REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	2000	1-7 July 1999 Final Report

4. TITLE AND SUBTITLE
International Conference on Curves and Surfaces (4th). Held in Saint-Malo, France on 1-7 July 1999. Proceedings, Volume 1. Curve and Surface Design.

5. FUNDING NUMBERS
F61775-99-WF068

6. AUTHOR(S)
Pierre-Jean Laurent, Paul Salonniere, Larry L. Schumaker, Editors

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Association Francaise d'Approximatiion (AFA)
c/o P.J. Laurent
La Pereree 38420
France

8. PERFORMING ORGANIZATION REPORT NUMBER
ISBN 0-8265-1356-5

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
EOARD
PSC 802 Box 14
FPO AE 09499-0039

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Published by Vanderbilt University Press, Nashville, TN. Series: Innovations in Applied Mathmatics. This work relates to Department of the Air Force grant issued by the European Aerospace Office of Research and Development. The United States has a royalty free license throughout the world in all copyrightable material contained herein.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for Public Release. U.S. Government Rights License. All other rights reserved by the copyright holder. (Code 1, 20)

12b. DISTRIBUTION CODE
A

12. ABSTRACT (Maximum 200 words)

This volume contains 45 papers presented at the 4th International Conference on Curves and Surfaces held in Saint-Malo, France on 1-7 July 1999. The companion Volume 2: Curve and Surface Fitting contains 43 papers.

The conference presents the classical domain of Approximation Theory (interpolation, smoothing techniques, splines, radial basis functions, wavelets) as well as more technical aspects of geometric modeling, computer-aided design and mechanics.

13. SUBJECT TERMS
EOARD, Foreign reports, Computer science, Computer graphics, Computer-aided design, Curve fitting, Mathematics

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

20011203 231

Curve and Surface Design

Saint-Malo 99

EDITED BY

Pierre-Jean Laurent
Laboratoire de Modélisation et Calcul
Université Joseph Fourier
Grenoble, France

Paul Sablonnière
Laboratoire de Logiciel, Analyse Numérique et Statistiques
Institut National des Sciences Appliquées
Rennes, France

Larry L. Schumaker
Department of Mathematics
Vanderbilt University
Nashville, Tennessee

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

VANDERBILT UNIVERSITY PRESS
Nashville

AQ F02-02-0268

U.S. Government Rights License
This work relates to Department of the Air Force Grant/rContract issued by the European Office Aerospace Research and Development (EOARD). The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein

© 2000 Vanderbilt University Press
All Rights Reserved

First Edition 2000

04 03 02 01 00 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Curve and surface design : Saint-Malo 99 / edited by Pierre-Jean Laurent, Paul Sablonnière, Larry L. Schumaker.
p. cm — (Innovations in applied mathematics)
Includes bibliographical references.
ISBN 0-8265-1356-5 (alk. paper)
1. Computer graphics. 2. Computer-aided design. 3. Curves, Algebraic—Data processing. 4. Surfaces, Algebraic—Data processing. I. Laurent, Pierre-Jean, 1937– II. Sablonnière, Paul, 1947– III. Schumaker, Larry L., 1939– IV. Series.

T385 .C857 2000
006.6'9—dc21 99-086571

Manufactured in the United States of America

Curve and Surface Design

Innovations in Applied Mathematics

An international series devoted to the latest research in modern areas of mathematics, with significant applications in engineering, medicine, and the sciences.

SERIES EDITOR:
Larry L. Schumaker
Stevenson Professor of Mathematics
Vanderbilt University

PREVIOUSLY PUBLISHED TITLES INCLUDE

Curves and Surfaces with Applications in CAGD (1997)

Surface Fitting and Multiresolution Methods (1997)

Mathematical Methods for Curves and Surfaces II (1998)

Mathematical Models in Medical and Health Science (1998)

Approximation Theory IX Volume 1. Theoretical Aspects (1999)

Approximation Theory IX Volume 2. Computational Aspects (1999)

CONTENTS

Preface viii

Contributors ix

Factorizations of Normalized Totally Positive Systems
J. M. Carnicer and E. Mainar 1

Intersections et Convergence
Paul de Faget de Casteljau 9

Implicitization Matrices in the Style of Sylvester with the Order of Bézout
Eng-Wee Chionh, Ming Zhang, and Ronald Goldman 17

Better Pasting via Quasi-Interpolation
Blair Conrad and Stephen Mann 27

Conjugate Silhouette Nets
W. L. F. Degen 37

From PS–splines to NURPS
Paul Dierckx and Joris Windmolders 45

Distance Calculation between a Point and a NURBS Surface
Eva Dyllong and Wolfram Luther 55

Curves from Motion, Motion from Curves
Rida T. Farouki 63

Curvature and Tangency Handles for Control of Convex Cubic Shapes
G. Figueroa, M. Paluszny, and F. Tovar 91

Monotonicity Conditions of Curvature for Bézier-de Casteljau Curves
Jean-Charles Fiorot and Laurent Schiavon 99

From Local Approximation to a G^1 Global Representation
Cédric Gérot, Dominique Attali, and Annick Montanvert 109

A Class of Totally Positive Blending B-Bases
Laura Gori, Laura Pezza, and Francesca Pitolli 119

Rational Ruled Surfaces Passing through Two Fixed Lines
Gueorgui H. Gueorguiev 127

Deformation Criteria for the Direct Manipulation of Free Form Surfaces
Stéphane Guillet and Jean-Claude Léon 135

Generation of Surfaces with Smooth Highlight Lines
Masatake Higashi, Hiroto Harada, and Mitsuru Kuroda 145

MIPS: An Efficient Global Parametrization Method
Kai Hormann and Günther Greiner 153

Comparison of Different Multisided Patches Using Algebraic Geometry
Kęstutis Karčiauskas and Rimvydas Krasauskas 163

N-sided Surface Generation from Arbitrary Boundary Edges
Kiyotaka Kato 173

Advantages of Topological Tools in Localization Methods
Mohammed Khachan and Patrick Chenin 183

Fast Voronoi Diagrams and Offsets on Triangulated Surfaces
Ron Kimmel and James A. Sethian 193

On the Geometry of Texture
Ron Kimmel, Nir A. Sochen, and Ravi Malladi 203

Optimization of a Curve Approximation Based on NURBS Interpolation
Jérôme Lépine, François Guibault, Marie-Gabrielle Vallet and Jean-Yves Trépanier 213

Filling N-sided Holes Using Combined Subdivision Schemes
Adi Levin 221

Dealing with Topological Singularities in Volumetric Reconstruction
H. Lopes, L. G. Nonato, S. Pesco, and G. Tavares 229

Linear Envelopes for Uniform B–spline Curves
David Lutterkort and Jörg Peters 239

N-sided Surfaces: a Survey
Pierre Malraison 247

Approximated Planes in Parallel Coordinates
Tanya Matskewich, Alfred Inselberg and Michel Bercovier 257

Applications of Sphere Geometry in Canal Surface Design
Christoph Mäurer 267

Geometric Computing with CGAL and LEDA
Kurt Mehlhorn and Stefan Schirra 277

A Geometric Approach for Knot Selection in Convexity-Preserving Spline Approximation
R. Morandi, D. Scaramelli, and A. Sestini 287

Convergence of Approximations for Arrangements of Curves
Manuela Neagu and Bernard Lacolle 297

The (2-5-2) Spline Function
Jae H. Park and Leonard A. Ferrari 307

Error Analysis of Algorithms for Evaluating Bernstein-Bézier-Type Multivariate Polynomials
J. M. Peña 315

A Basis for Homogeneous Polynomial Solutions to Homogeneous Constant Coefficient PDE's: An Algorithmic Approach Through Apolarity
Michel Pocchiola and Gert Vegter 325

Triangular G^2-Splines
Hartmut Prautzsch and Georg Umlauf 335

On Calculating with Lower Order Chebyshev Splines
Mladen Rogina and Tina Bosner 343

A Declarative Modeler for B-Spline Curves
Vincent Rossignol and Marc Daniel 353

Diffuse Curvature Computation for Surface Recognition
J. M. Savignat, O. Stab, A. Rassineux, and P. Villon 363

Discrete Fairing of Curves and Surfaces Based on Linear Curvature Distribution
R. Schneider and L. Kobbelt 371

Triangulating Trimmed NURBS Surfaces
Chang Shu and Pierre Boulanger 381

Surface Interpolation of Non-four-sided and Concave Area by NURBS Boundary Gregory Patches
Junji Sone, Kouichi Konno, and Hiroaki Chiyokura 389

Central Conics on Parabolic Dupin Cyclides
Kenji Ueda 399

Adaptive Parameterization and Approximation for CAD Data Reduction
G. Wahu, J. M. Brun, and A. Bouras 407

On the Geometry of Sculptured Surface Machining
Johannes Wallner and Helmut Pottmann 417

Extensions: Extrapolation Methods for CAD
Hans J. Wolters 433

PREFACE

During the week of July 1-7, 1999, the Fourth International Conference on *Curves and Surfaces* was held in Saint-Malo (France). It was organized by the *Association Française d'Approximation*, (A.F.A.). The organizing committee consisted of L. Amodei (Toulouse), J.-L. Bauchat (Metz), A. Cohen (Paris), J.-C. Fiorot (Valenciennes), J. Gaches (Toulouse), G.-P. Bonneau (Grenoble), Y. Lafranche (Rennes), P.-J. Laurent (Grenoble), M.-L. Mazure (Grenoble), J.-L. Merrien (Rennes), C. Potier (Paris), C. Rabut (Toulouse), P. Sablonnière (Rennes), L.L. Schumaker (Nashville), C. Vercken (Paris).

The conference was attended by 275 mathematicians from 37 different countries, and the program included 10 invited one-hour lectures and 190 half-hour research talks or poster presentations. A number of research talks were presented in eight minisymposia organized by W. Dahmen, R. DeVore, D. Donoho, J. Hoschek, B. Lacolle, H. Pottmann, M. Sabin, and J. Stöckler.

The proceedings of this conference consists of this volume (containing 45 papers), and the companion volume *Curve and Surface Fitting: Saint-Malo 1999* (containing 43 papers).

We would like to thank the following institutions for their financial or technical support and their contribution to the success of this conference: Ministère de l'Education Nationale, de la Recherche et de la Technologie; European Office of Aerospace Research and Development (Air Force Office of Scientific Research, United States Air Force Research Laboratory); Institut National des Sciences Appliquées de Rennes; Institut d'Informatique et de Mathématiques Appliquées de Grenoble; Conseil Régional de Bretagne; Ministère de la Défense (contrat No 9960014, Direction des Systèmes de Forces et de la Prospective, Service de la Recherche et des Etudes Amont, Sous-direction Scientifique, Bureau de la Prospective Scientifique, Délégation Générale pour l'Armement); Université Pierre et Marie Curie (Paris); Laboratoire de Modélisation et Calcul de Grenoble; Institut National des Sciences Appliquées de Toulouse; Université Joseph Fourier (Grenoble); Vanderbilt University (Nashville); Ministère des Affaires Etrangères; Matra Datavision; Ecole Nationale Supérieure des Arts et Métiers de Metz; France Télécom; Ecole Nationale Supérieure des Télécommunications (Paris); Ecole Centrale de Nantes.

We would like to thank Gerda Schumaker for assisting with the preparation of the proceedings.

Nashville, Tennessee April 5, 2000

CONTRIBUTORS

Numbers in parentheses indicate pages on which authors' contributions begin. Articles marked with * are in *Curve and Surface Design: Saint-Malo 1999*, and articles marked with ** are in *Curve and Surface Fitting: Saint-Malo 1999.*

GIAMPIETRO ALLASIA (**75), *Dipartimento di Matematica, Università di Torino, I-10123 Torino, Italy* [allasia@dm.unito.it]

IAIN J. ANDERSON (**1), *School of Computing and Mathematics, University of Huddersfield, Queensgate, Huddersfield, West Yorkshire, HD1 3DH, UK* [i.j.anderson@hud.ac.uk]

D. APPRATO (**9), *Université de Pau, Dpt de Mathématiques, IPRA, Av. de l'Université, 64000 Pau, France*

FRANCESC ARÀNDIGA (**19), *Departament de Matemàtica Aplicada, Universitat de València, 4610 Burjassot, València, Spain* [arandiga@uv.es]

DOMINIQUE ATTALI (*109), *Laboratoire LIS, ENSIEG, Domaine Universitaire, 38402 St-Martin-d'Hères, France* [Dominique.Attali@lis.inpg.fr]

SILVIA BACCHELLI (**27), *Dipartimento di Matematica Pura e Applicata, Università di Padova, Via Belzoni 7, Padova, Italy* [silvia@agamennone.csr.unibo.it]

EDUARDO ANTÔNIO BARROS DA SILVA (**153), *DEL/PEE/EE/COPPE, Universidade Federal do Rio de Janeiro, Brazil* [eduardo@lps.ufrj.br]

BRIAN A. BARSKY (**393), *Computer Science Division, University of California, Berkeley, CA 94720-1776, USA* [barsky@cs.berkeley.edu]

RICK BEATSON (**37,**47), *Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand* [R.Beatson@math.canterbury.ac.nz]

MOHAMMED-NAJIB BENBOURHIM (**57), *Université Paul Sabatier (M.I.P), 118, Route de Narbonne, 31062 Toulouse Cedex 04, France* [bbourhim@cict.fr]

MICHEL BERCOVIER (*257), *The Hebrew University of Jerusalem, Givat Ram Campus, Institute of Computer Science, 991904 Jerusalem, Israel* [berco@cs.huji.ac.il]

STEFANO BERRONE (**65), *Dipartimento di Ingegneria Aeronautica e Spaziale, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Torino, Italy* [sberrone@calvino.polito.it]

RENATA BESENGHI (**75), *Dipartimento di Matematica, Università di Torino, I-10123 Torino, Italy* [besenghi@dm.unito.it]

J. B. BETBEDER (**9), *Université de Pau, Dpt de Mathématiques, IPRA, Av. de l'Université, 64000 Pau, France*

GEORGES–PIERRE BONNEAU (**209,**237), *LMC–IMAG, Université Joseph Fourier, BP 53, 38041 Grenoble Cedex 9, France* [Georges-Pierre.Bonneau@imag.fr]

TINA BOSNER (*343), *Dept. of Mathematics, University of Zagreb, Bijenička 30, 10000 Zagreb, Croatia*

PIERRE BOULANGER (*381), *Institute for Information Technology, National Research Council of Canada, Montreal Road, Ottawa, Ontario, Canada K1A 0R6* [pierre.boulanger@iit.nrc.ca]

A. BOURAS (*407), *LIGIM, University Claude Bernard Lyon 1, 43 boulevard du 11 novembre 1918, 69622 Villeurbanne, France*

MIRA BOZZINI (**85), *Dipartimento di Matematica e Applicazioni, Università di Milano Bicocca, via Bicocca degli Arcimboldi 8, 20126 Milano, Italy* [bozzini@matapp.unimib.it]

PETER BROCKWELL (**343), *Department of Statistics, Colorado State University, Ft. Collins, CO 80523-1877 USA* [pjbrock@stat.colostate.edu]

J. M. BRUN (*407), *LIM/XAOLab, University Aix-Marseille 2, ESIL, Case 925, 13288 Marseille Cedex 9, France* [jmbrun@esil.univ-mrs.fr]

EMMANUEL CANDES (**95,**105), *Department of Statistics, Stanford University, Stanford, CA 94305-4065, USA* [emmanuel@stat.stanford.edu]

J. M. CARNICER (*1), *Departamento de Matemática Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain* [carnicer@posta.unizar.es]

PAUL DE FAGET DE CASTELJAU (*9), *4 Avenue du Commerce, 78000 Versailles, France*

E. CHACKO (**37), *Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand* [E.Chacko@math.canterbury.ac.nz]

PATRICK CHENIN (*183), *LMC-IMAG, Université Joseph Fourier, BP 53, 38.041 Grenoble Cedex 9, France* [pchenin@imag.fr]

J. B. CHERRIE (**47), *Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand* [J.Cherrie@math.canterbury.ac.nz]

ENG-WEE CHIONH(*17), *School of Computing, National University of Singapore, Lower Kent Ridge Road, Singapore 119260* [chionhew@comp.nus.edu.sg]

H. CHIYOKURA (*389), *Faculty if Environmental Information, Keio University, 5322 Endo, Fujisawa, Kanagawa, 252-0816, Japan* [chiyo@sfc.keio.ac.jp]

ZBIGNIEW CIESIELSKI (**121), *Instytut Matematyczny PAN, ul. Abrahama 18, 81-825 Sopot, Poland* [Z.Ciesielski@impan.gda.pl]

BLAIR CONRAD (*27), *Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1* [beconrad@cgl.uwaterloo.ca]

C. CONTI (**135), *Dipartimento di Energetica, Università di Firenze, Via C. Lombroso 6/17, I–50134 Firenze, Italy* [costanza@sirio.de.unifi.it]

PAOLO COSTANTINI (**143), *Dipartimento di Matematica " Roberto Magari", Via del Capitano 15, 53100 Siena, Italy* [costantini@unisi.it]

MARIANTONIA COTRONEI (**27), *Dipartimento di Matematica, Università di Messina, Salita Sperone, 31, Messina, Italy* [marianto@dipmat.unime.it]

MARCOS CRAIZER (**153), *Departamento de Matemática, Pontifícia Universidade Católica do Rio de Janeiro, R. Marquês de São Vicente, 225 - Gávea, Rio de Janeiro, RJ, 22453-900 Brazil* [craizer@mat.puc-rio.br]

MARC DANIEL (*353), *XAOlab-ESIL/LIM, Campus de Luminy, case postale 925, 13288 Marseille Cedex 9, France* [Marc.Daniel@esil.univ-mrs.fr]

OLEG DAVYDOV (**161,**171), *Justus-Liebig-Universität Giessen, Mathematisches Institut, 35392 Giessen, Germany* [oleg.davydov@math.uni-giessen.de]

LUBOMIR T. DECHEVSKI (**181), *Département de mathématiques et statistique, Université de Montréal, C.P. 6128, Succursale A, Montréal (Québec) H3C 3J7, Canada* [dechevsk@dms.umontreal.ca]

HAFSA DEDDI (**191), *Department of Mathematics and Computer Science, University of Lethbridge, 4401 University Drive, Lethbridge, Alberta, T1K 3M4, Canada* [deddi@cs.uleth.ca]

W. L. F DEGEN (*37), *University of Stuttgart, Mathematisches Institut B, Pfaffenwaldring 57, D-70569 Stuttgart, Germany* [degen@mathematik.uni-stuttgart.de]

PAUL DIERCKX (*45), *Celestijnenlaan 200A, 3001 Heverlee, Belgium* [paul.dierckx@cs.kuleuven.ac.be]

FABRICE DODU (**201), *Institut National des Sciences Appliquées de Toulouse, U.M.R. 9974 C.N.R.S./I.N.S.A./M.I.P., Génie Mathématique*

et Modélisation, 135 Avenue de Rangueil, 31077 Toulouse Cedex 4, France [dodu@gmm.insa-tlse.fr]

ROSA DONAT (**19), *Departament de Matemàtica Aplicada, Universitat de València, 4610 Burjassot. València, Spain* [donat@uv.es]

DAVID DONOHO (**105), *Department of Statistics, Stanford University, Stanford, CA 94305-4065, USA* [donoho@stat.stanford.edu]

EVA DYLLONG (*55), *Dept. of Computer Science, University of Duisburg, D-47048 Duisburg, Germany* [dyllong@informatik.uni-duisburg.de]

HAZEL EVERETT (**191), *LORIA - INRIA Lorraine, 615 rue du jardin botanique, B.P. 101, 54602 Villers-les-Nancy Cedex, France* [everett@loria.fr]

RIDA T. FAROUKI (*63), *Department of Mechanical and Aeronautical Engineering, University of California, Davis, CA 95616, USA* [farouki@ucdavis.edu]

LEONARD A. FERRARI (*307), *The Bradley Department of Electrical Computer Engineering, 340 Whittemore Hall, Virginia Tech, Blacksburg, VA 24061-0111, USA* [ferrari@vt.edu]

G. FIGUEROA (*91), *Universidad Central de Venezuela. Facultad de Ciencias, Escuela de Matemática, Laboratorio de Computación Gráfica y Geometría Aplicada, Apartado 47809, Los Chaguaramos, Caracas 1041-A, Venezuela* [giovanni@euler.ciens.ucv.ve]

JEAN-CHARLES FIOROT (*99), *Université de Valenciennes et du Hainaut-Cambrésis, Laboratoire MACS, B.P.311, 59304 Valenciennes Cedex, France* [fiorot@univ-valenciennes.fr]

DÉCIO ANGELO FONINI JR. (**153), *PEE/COPPE, Universidade Federal do Rio de Janeiro, Brazil* [fonini@lps.ufrj.br]

CÉDRIC GÉROT (*109), *Laboratoire LIS, ENSIEG, Domaine Universitaire, 38402 St-Martin-d'Hères, France* [Cedric.Gerot@lis.inpg.fr]

ALEXANDRE GERUSSI (**209), *LMC–IMAG, Université Joseph Fourier, BP 53, 38041 Grenoble Cedex 9, France* [Alexandre.Gerussi@imag.fr]

RONALD GOLDMAN (*17), *Department of Computer Science, Rice University, Houston, TX 77025, USA* [rng@cs.rice.edu]

TIM N. T. GOODMAN (**219), *Dept. of Mathematics, The University, Dundee DD1 4HN, U.K.* [tgoodman@mcs.dundee.ac.uk]

LAURA GORI (*119), *Dept. MeMoMat, Università La Sapienza, Via Antonio Scarpa 16 - 00161, Roma, Italy* [gori@dmmm.uniroma1.it]

A. ARDESHIR GOSHTASBY (**227), *Department of Computer Science and Engineering, Wright State University, Dayton, OH 45435, USA* [agoshtas@cs.wright.edu]

C. GOUT (**9), *Université de Pau, Dpt de Mathématiques, IPRA, Av. de l'Université, 64000 Pau, France*
[chris_gout@alum.calberkeley.org]

GÜNTHER GREINER (*153), *Computer Graphics Group, University of Erlangen-Nürnberg, Am Weichselgarten 9, D-91058 Erlangen, Germany*
[greiner@informatik.uni-erlangen.de]

GUEORGUI H. GUEORGUIEV (*127), *Faculty of Mathematics and Informatics, Shumen University, 9712 Shumen, Bulgaria*
[g.georgiev@shu-bg.net]

FRANÇOIS GUIBAULT (*213), *Département de génie électrique et de génie informatique, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-Ville, Montréal (Québec), H3C 3A7, Canada*
[francois@cerca.umontreal.ca]

STÉPHANE GUILLET (*135), *Laboratoire 3S, Domaine Universitaire, BP 53X, 38041 Grenoble Cedex 9, France*
[Stephane.Guillet@hmg.inpg.fr]

STEFANIE HAHMANN (**237), *Laboratoire de Modélisation et Calcul, CNRS-IMAG, B.P. 53, F-38041 Grenoble Cedex 9, France*
[hahmann@imag fr]

STEPHEN HALES (**247), *Mathematics & Computer Science Dept., University of Leicester, University Road, Leicester, LE1 7RH, U. K.*
[sjh16@mcs.le.ac.uk]

HIROTO HARADA (*145), *Dept. of Mechanical Systems Engineering, Toyota Technological Institute, 2-12-1, Hisakata, Tempaku-ku, Nagoya 468-8511, Japan* [920043@nhkspg.co.jp]

MASATAKE HIGASHI (*145), *Dept. of Mechanical Systems Engineering, Toyota Technological Institute, 2-12-1, Hisakata, Tempaku-ku, Nagoya 468-8511, Japan* [higahsi@toyota-ti.ac.jp]

KAI HORMANN (*153), *Computer Graphics Group, University of Erlangen-Nürnberg, Am Weichselgarten 9, D-91058 Erlangen, Germany*
[hormann@informatik.uni-erlangen.de]

ALFRED INSELBERG (*257), *School of Mathematical Sciences, Tel Aviv University, Tel Aviv 61396, Israel* [aiisreal@cs.tau.ac.il]

KURT JETTER (**135), *Institut für Angew. Math. und Statistik, Universität Hohenheim, D–70593 Stuttgart, Germany*
[kjetter@uni-hohenheim.de]

BERT JÜTTLER (**385), *Darmstadt University of Technology, Dept. of Mathematics, Schloßgartenstr. 7, 64289 Darmstadt, Germany*
[juettler@mathematik.tu-darmstadt.de]

KĘSTUTIS KARČIAUSKAS (*163), *Dept. of Mathematics and Computer Science, Vilnius University, Naugarduko 24, 2600 Vilnius, Lithuania* [kestutis.karciauskas@maf.vu.lt]

KIYOTAKA KATO (*173), *Mitsubishi Electric Corporation, Industrial Electronics and Systems Lab, 8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, Japan* [kato@sdl.melco.co.jp]

MOHAMMED KHACHAN (*183), *L.E.R.I (Laboratoire d'Etudes et de Recherches Informatiques), IUT-Léonard de Vinci, 51.059 Reims, France* [khachan@leri.univ-reims.fr]

RON KIMMEL (*193,*203), *Computer Science Department, Technion, Israel Institute of Technology, Haifa 32000, Israel* [ron@cs.technion.ac.il]

LEIF KOBBELT (*371), *Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany* [kobbelt@mpi-sb.mpg.de]

KOUICHI KONNO (*389), *RICOH corporation, 1-1-17 Koishikawa, Bunkyou, Tokyo, 112-0002, Japan* [konno@src.ricoh.co.jp]

V. V. KOVTUNETS (**255), *blvrd. Lesi Ukrainky 21a, ap.17, Kyiv, Ukraine 01133* [ktp@carrier.kiev.ua]

JERNEJ KOZAK (**263), *FMF and IMFM, University of Ljubljana, Jadranska 19, 1000 Ljubljana, Slovenia* [Jernej.Kozak@FMF.Uni-Lj.Si]

RIMVYDAS KRASAUSKAS (*163), *Dept. of Mathematics and Computer Science, Vilnius University, Naugarduko 24, 2600 Vilnius, Lithuania* [rimvydas.krasauskas@maf.vu.lt]

MITSURU KURODA (*145,**273), *Dept. of Mechanical Systems Engineering, Toyota Technological Institute, 2-12-1, Hisakata, Tempaku-ku, Nagoya 468-8511, Japan* [kuroda@toyota-ti.ac.jp]

BERNARD LACOLLE (*297), *LMC-IMAG, BP 53, 38041 Grenoble Cedex 9, France* [Bernard.Lacolle@imag.fr]

SYLVAIN LAZARD (**191), *LORIA - INRIA Lorraine, 615 rue du jardin botanique, B.P. 101, 54602 Villers-les-Nancy Cedex, France* [lazard@loria.fr]

DAMIANA LAZZARO (**27), *Dipartimento di Matematica, Università di Bologna, Piazza di Porta S. Donato, 5, Bologna, Italy* [lazzaro@csr.unibo.it]

MICHEL LÉGER (**281), *1 & 4 avenue de Bois-Préau, 92852 Rueil-Malmaison, France* [michel.leger@ifp.fr]

JEAN-CLAUDE LÉON (*135), *Laboratoire 3S, Domaine Universitaire, BP 53X, 38041 Grenoble Cedex 9, France* [Jean-Claude.Leon@hmg.inpg.fr]

JÉRÔME LÉPINE (*213), *Département de génie mécanique, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-Ville, Montréal (Québec), H3C 3A7, Canada* [lepine@cerca.umontreal.ca]

JEREMY LEVESLEY (**247,**291), *Mathematics & Computer Science Dept., University of Leicester, University Road, Leicester, LE1 7RH, U. K.* [jl1@mcs.le.ac.uk]

ADI LEVIN (*221), *Tel-Aviv University, School of Mathematical Sciences, Tel-Aviv 69978, Israel* [adilev@inter.net.il]

KIA-FOCK LOE (**393), *Department of Computer Science, School of Computing, National University of Singapore, Lower Kent Ridge, Singapore 119260* [loekf@comp.nus.edu.sg]

H. LOPES (*229), *Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, Brazil, CEP:22.453-900* [lopes@mat.puc-rio.br]

LIN-TIAN LUH (**301), *Dept. of Applied Mathematics, Providence University, 200 Chungchi Road, Shalu 43301, Taichung Hsien, Taiwan, Republic of China* [ltluh@pu.edu.tw]

WOLFRAM LUTHER (*55), *Dept. of Computer Science, University of Duisburg, D-47048 Duisburg, Germany* [luther@informatik.uni-duisburg.de]

DAVID LUTTERKORT (*239), *Dept. Computer Sciences, Purdue University, West Lafayette, IN, 47905-1398 USA* [lutter@cise.ufl.edu]

YVON MADAY (**309), *Laboratoire d'Analyse Numérique, Université Pierre et Marie Curie, 4, Place Jussieu, 75252, Paris Cedex 05, France* [maday@ann.jussieu.fr]

E. MAINAR (*1), *Departamento de Matemática Aplicada, Universidad de Zaragoza, 50009 Zaragoza, Spain* [esme@posta.unizar.es]

RAVI MALLADI (*203), *Lawrence Berkeley National Laboratory, University of California, Berkeley, CA 94720, USA* [malladi@math.lbl.gov]

PIERRE MALRAISON (*247), *Spatial Technology, Inc., 2425 55th Street, Suite 100, Boulder, CO 80301 USA* [pierre_malraison@spatial.com]

STEPHEN MANN (*27), *Computer Science Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1* [smann@cgl.uwaterloo.ca]

CARLA MANNI (**143), *Dipartimento di Matematica, Via Carlo Alberto 10, 10123 Torino, Italy* [manni@dm.unito.it]

J. C. MASON (**1), *School of Computing and Mathematics, University of Huddersfield, Queensgate, Huddersfield, West Yorkshire, HD1 3DH, UK* [j.c.mason@hud.ac.uk]

TANYA MATSKEWICH (*257), *The Hebrew University of Jerusalem, Givat Ram Campus, Institute of Computer Science, 991904 Jerusalem, Israel* [fisa@cs.huji.ac.il]

CHRISTOPH MÄURER (*267), *TWT-GmbH, Finkenstr. 16, 70794 Filderstadt, Germany* [fa-twt.christoph.maeurer@daimlerchrysler.com]

KURT MEHLHORN (*277), *Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany* [mehlhorn@mpi-sb.mpg.de]

H. MICHAEL MÖLLER (**325,**333), *Fachbereich Mathematik, der Universität, 44221 Dortmund, Germany* [hmm@mathematik.uni-dortmund.de]

ANNICK MONTANVERT (*109), *Laboratoire LIS, ENSIEG, Domaine Universitaire, 38402 St-Martin-d'Hères, France* [Annick.Montanvert@lis.inpg.fr]

ROSSANA MORANDI (*287), *Dip. di Energetica "S. Stecco", Via Lombroso 6/17, 50134 Firenze, Italy* [morandi@de.unifi.it]

SHINJI MUKAI (**273), *Maebashi Institute of Technology, 460-1 Kamisanaru, Maebashi, Gunma 371-0816, Japan* [mukai@maebashi-it.ac.jp]

PHILIPPE NAVEAU (**343), *Geophysical Statistics Project, National Center for Atmospheric Research, Boulder, CO 80307 B.P. Box 3000 USA* [pnaveau@ucar.edu]

MANUELA NEAGU (*297), *LMC-IMAG, BP 53, 38041 Grenoble Cedex 9, France* [Manuela.Neagu@imag.fr]

L. NONATO (*229), *Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, Brazil, CEP:22.453-900* [nonato@mat.puc-rio.br]

GÜNTHER NÜRNBERGER (**161), *Universität Mannheim, Fakultät für Mathematik und Informatik, A5, 68131 Mannheim, Germany* [nuern@euklid.math.uni-mannheim.de]

DOUG NYCHKA (**343), *Geophysical Statistics Project, National Center for Atmospheric Research, Boulder, CO 80307 B.P. Box 3000 USA* [nychka@ucar.edu]

M. PALUSZNY (*91), *Universidad Central de Venezuela. Facultad de Ciencias, Escuela de Matemática, Laboratorio de Computación Gráfica y Geometría Aplicada. Apartado 47809, Los Chaguaramos, Caracas 1041-A, Venezuela* [marco@euler.ciens.ucv.ve]

JAE H. PARK (*307), *The Bradley Department of Electrical Computer Engineering, 340 Whittemore Hall, Virginia Tech, Blacksburg, VA 24061-0111, USA* [park@vt.edu]

J. M. PEÑA (*315), *Departamento de Matemática Aplicada, Universidad de Zaragoza, 50006 Zaragoza, Spain* [jmpena@posta.unizar.es]

S. PESCO (*229), *Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, Brazil, CEP:22.453-900* [pesco@mat.puc-rio.br]

MARTIN PETERNELL (**351), *Institut für Geometrie, TU Wien, Wiedner Hauptstrasse 8-10, A-1040 Wien, Austria* [peternell@geometrie.tuwien.ac.at]

JÖRG PETERS (*239), *Dept C.I.S.E., CSE Bldg, University of Florida, Gainesville, FL 32611-6120, USA* [jorg@cise.ufl.edu]

LAURA PEZZA (*119), *Dept. MeMoMat, Università La Sapienza, Via Antonio Scarpa 16 - 00161, Roma, Italy* [pezza@dmmm.uniroma1.it]

FRANCESCA PITOLLI (*119), *Dept. MeMoMat, Università La Sapienza, Via Antonio Scarpa 16 - 00161, Roma, Italy* [pitolli@dmmm.uniroma1.it]

MICHEL POCCHIOLA (*325), *Département d'Informatique, Ecole Normale Supérieure, 45, rue d'Ulm, 75230 Paris Cedex 05 - France* [Michel.Pocchiola@dmi.ens.fr]

HELMUT POTTMANN (*417,**351), *Institut für Geometrie, Wiedner Hauptstraße 8–10/113, A-1040 Wien, Austria* [pottmann@geometrie.tuwien.ac.at]

HARTMUT PRAUTZSCH (*335), *Institute for Operating- and Dialogsystems, Universität Karlsruhe, D-76128 Karlsruhe, Germany* [prau@ira.uka.de]

DAVID L. RAGOZIN (**47,**291), *University of Washington, Box 354350, Department of Mathematics, Seattle, WA 98195-4350 USA* [rag@math.washington.edu]

ALAIN RASSINEUX (*363), *Université de Technologie de Compiegne, U.R.A. CNRS 1505 LG2MS, BP 529, F-60205 Compiegne, Cedex, France* [alain.rassineux@utc.fr]

MLADEN ROGINA (*343), *Dept. of Mathematics, University of Zagreb, Bijenička 30, 10000 Zagreb, Croatia* [rogina@math.hr]

COLIN ROSS (**1), *School of Computing and Mathematics, University of Huddersfield, Queensgate, Huddersfield, West Yorkshire, HD1 3DH, UK* [c.ross@hud.ac.uk]

VINCENT ROSSIGNOL (*353), *IRIN (Institut de Recherche en Informatique de Nantes), 2, Rue de la Houssinière, BP 92208, 44322 Nantes Cedex 3, France* [Vincent.Rossignol@irin.univ-nantes.fr]

MILVIA ROSSINI (**85), *Dipartimento di Matematica e Applicazioni, Università di Milano Bicocca, via Bicocca degli Arcimboldi 8, 20126 Milano, Italy* [rossini@matapp.unimib.it]

THOMAS SAUER (**325,**333), *Mathematisches Institut, der Universität Erlangen–Nürnberg, Bismarckstr. 1 1/2, 91054 Erlangen, Germany* [sauer@mi.uni-erlangen.de]

J. M. SAVIGNAT (*363), *École des Mines, 35, rue Saint Honoré, 77 305 Fontainebleau Cedex, France* [savignat@cges.ensmp.fr]

DANIELA SCARAMELLI (*287), *Dip. di Matematica Pura ed Applicata, Via G. Belzoni 7, 35131 Padova, Italy* [daniela@sirio.de.unifi.it]

ROBERT SCHABACK (**359), *Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestraße 16-18, D-37083 Göttingen, Germany* [schaback@math.uni-goettingen.de]

KARL SCHERER (**375), *Institut für Angewandte Mathematik, Universität Bonn, Bonn, 53115 Germany* [scherer@iam.uni-bonn.de]

LAURENT SCHIAVON (*99), *Université de Valenciennes et du Hainaut-Cambrésis, Laboratoire MACS, B.P.311, 59304 Valenciennes Cedex, France* [schiavon@univ-valenciennes.fr]

STEFAN SCHIRRA (*277), *Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany* [stschirr@mpi-sb.mpg.de]

ROBERT SCHNEIDER (*371), *Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany* [schneider@mpi-sb.mpg.de]

LARRY L. SCHUMAKER (**171), *Department of Mathematics, Nashville, TN 37240, USA* [s@mars.cas.vanderbilt.edu]

U. SCHWANECKE (**385), *Darmstadt University of Technology, Dept. of Mathematics, Schloßgartenstr. 7, 64289 Darmstadt, Germany* [schwanecke@mathematik.tu-darmstadt.de]

ALESSANDRA SESTINI (*287), *Dip. di Energetica "S. Stecco", Via Lombroso 6/17, 50134 Firenze, Italy* [sestini@de.unifi.it]

JAMES A. SETHIAN (*193), *Department of Mathematics, and Lawrence Berkeley National Laboratory, University of California, Berkeley, CA 94720, USA* [sethian@math.berkeley.edu]

CHANG SHU (*381), *Institute for Information Technology, National Research Council of Canada, Montreal Road, Ottawa, Ontario, Canada K1A 0R6* [chang.shu@iit.nrc.ca]

NIR A. SOCHEN (*203), *School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel* [sochen@math.tau.ac.il]

JUNJI SONE (*389), *Toshiba corporation, 1 Toshiba-tyou, Futyuu, Tokyo, 183-0043, Japan* [junji.sone@toshiba.co.jp]

OLIVIER STAB (*363), *Ecole des Mines - C.G.E.S, 35, rue saint Honore, 77 305 Fontainebleau Cedex, France* [stab@cges.ensmp.fr]

CHIEW-LAN TAI (**393), *Department of Computer Science, The Hong Kong University of Science & Technology, Clear Water Bay, Kowloon, Hong Kong* [taicl@cs.ust.hk]

RIADH TALEB (**237), *Laboratoire de Modélisation et Calcul, CNRS-IMAG, B.P. 53, F-38041 Grenoble Cedex 9, France* [talebr@imag.fr]

G. TAVARES (*229), *Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Matemática, Rua Marquês de São Vicente, 225, Gávea, Rio de Janeiro, RJ, Brazil, CEP:22.453-900* [geovan@mat.puc-rio.br]

HOLGER THEISEL (**403), *University of Rostock, Computer Science Department, PostBox 999, 18051 Rostock, Germany* [theisel@informatik.uni-rostock.de]

F. TOVAR (*91), *Universidad Central de Venezuela. Facultad de Ciencias, Escuela de Matemática, Laboratorio de Computación Gráfica y Geometría Aplicada. Apartado 47809, Los Chaguaramos, Caracas 1041-A, Venezuela* [ftovar@euler.ciens.ucv.ve]

JEAN-YVES TRÉPANIER (*213), *Département de génie mécanique, École Polytechnique de Montréal, C.P. 6079, Succ. Centre-Ville, Montréal (Québec), H3C 3A7, Canada* [jyves@cerca.umontreal.ca]

KENJI UEDA (*399), *Ricoh Company, Ltd., 1-1-17, Koishikawa, Bunkyo-ku, Tokyo 112-0002, Japan* [ueda@src.ricoh.co.jp]

GEORG UMLAUF (*335), *Dept. C.I.S.E., CSE Bldg, University of Florida, Gainesville, FL 32611-6120, USA* [umlauf@cise.ufl.edu]

KARSTEN URBAN (**65), *Institut für Geometrie und Praktische Mathematik, RWTH Aachen, Templergraben 55, 52056 Aachen, Germany* [urban@igpm.rwth-aachen.de]

MARIE-GABRIELLE VALLET (*213), *CERCA, 5160 boul. Decarie, #400, Montréal (Québec), H3X 2H9, Canada* [vallet@cerca.umontreal.ca]

GERT VEGTER (*325), *Department of Mathematics and Computing Science, University of Groningen, P. O. Box 800, 9700 AV Groningen, The Netherlands* [gert@cs.rug.nl]

S. VEIRA-TESTE (**9), *TOPCAD SA, BP 521, 31674 Labege, France*

PIERRE VILLON (*363), *Université de Technologie de Compiegne, U.R.A. CNRS 1505 LG2MS, BP 529, F-60205 Compiegne, Cedex, France* [pierre.villon@utc.fr]

G. WAHU (*407), *LIGIM, University Claude Bernard Lyon 1, 43 boulevard du 11 novembre 1918, 69622 Villeurbanne, France* [gwahu@ligim.univ-lyon1.fr]

JOHANNES WALLNER (*417), *Institut für Geometrie, Wiedner Hauptstraße 8–10/113, A-1040 Wien, Austria* [wallner@geometrie.tuwien.ac.at]

JOE WARREN (**411), *Rice University, Department of Computer Science, P. O. Box 1892, Houston, TX 77251-1892, USA* [jwarren@rice.edu]

HENRIK WEIMER (**411), *Rice University, Department of Computer Science, P.O. Box 1892, Houston, TX 77251-1892, USA* [henrik@rice.edu]

HOLGER WENDLAND (**359), *Institut für Numerische und Angewandte Mathematik, Universität Göttingen, Lotzestraße 16-18, D-37083 Göttingen, Germany* [wendland@math.uni-goettingen.de]

WOLFGANG L. WENDLAND (**181), *Mathematisches Institut A, Universität Stuttgart, Pfaffenwaldring 57, 70569 Stuttgart, Germany* [wendland@mathematik.uni-stuttgart.de]

JORIS WINDMOLDERS (*45), *Celestijnenlaan 200A, 3001 Heverlee, Belgium* [joris.windmolders@cs.kuleuven.ac.be]

HANS J. WOLTERS (*433), *Hewlett Packard Laboratories, Palo Alto, CA, USA* [wolters@hpl.hp.com]

EMIL ŽAGAR (**263), *FRI and IMFM, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenia* [Emil@Gollum.Fri.Uni-Lj.Si]

FRANK ZEILFELDER (**161), *Universität Mannheim, Fakultät für Mathematik und Informatik, A5, 68131 Mannheim, Germany* [zeilfeld@fourier.math.uni-mannheim.de]

MING ZHANG (*17), *Department of Computer Science, Rice University, Houston, TX 77025, USA* [mzhang@cs.rice.edu]

Factorizations of Normalized Totally Positive Systems

J. M. Carnicer and E. Mainar

Abstract. The de Casteljau algorithm for evaluation of Bézier curves can be generalized to curves generated by any normalized totally positive basis. The construction of this algorithm is based upon a factorization of the system as a product of bidiagonal stochastic matrices of functions. These factorizations depend on a selection of a sequence of rectangular bidiagonal matrices of decreasing dimensions.

§1. Introduction

The Bernstein basis $b_i^n(t) := \binom{n}{i}(1-t)^{n-i}t^i$ can be used for defining a Bézier curve

$$\gamma(t) := \sum_{i=0}^{n} P_i b_i^n(t), \quad t \in [0,1].$$

By means of the degree raising technique, we can express the Bézier curve in terms of the Bernstein basis of one higher degree: $\gamma(t) = \sum_{i=0}^{n+1} Q_i b_i^{n+1}(t)$, $t \in [0,1]$. Indeed, the relations

$$b_i^n(t) = \frac{n-i+1}{n+1} b_i^{n+1}(t) + \frac{i+1}{n+1} b_{i+1}^{n+1}(t), \quad i = 0, \dots, n, \tag{1.1}$$

can be written in matrix form as

$$(b_0^n, \dots, b_n^n) = (b_0^{n+1}, \dots, b_{n+1}^{n+1})A, \tag{1.2}$$

where A is an $(n+2)\times(n+1)$ nonnegative stochastic bidiagonal matrix. Such a matrix can be written as:

$$A = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \alpha_1 & 1-\alpha_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & 1-\alpha_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \alpha_n & 1-\alpha_n \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}, \quad 0 \le \alpha_i \le 1, \qquad i = 1, \dots, n. \tag{1.3}$$

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 1–8.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

Equality (1.2) corresponds to the choice

$$\alpha_i := \frac{i}{n+1}, \quad i = 1, \dots, n. \tag{1.4}$$

Using (1.2), we can write

$$\gamma(t) = (b_0^n, \dots, b_n^n)(P_0, \dots, P_n)^T = (b_0^{n+1}, \dots, b_{n+1}^{n+1})A(P_0, \dots, P_n)^T,$$

which proves that the new control polygon is given by

$$(Q_0, \dots, Q_{n+1})^T := A(P_0, \dots, P_n)^T. \tag{1.5}$$

On the other hand, the de Casteljau algorithm for the pointwise evaluation of the curve is based on the following well-known recurrence relations

$$b_i^{n+1}(t) = \lambda_{i-1}(t)b_{i-1}^n(t) + (1 - \lambda_i(t))b_i^n(t), \quad i = 0, \dots, n+1, \tag{1.6}$$

where $\lambda_i(t) := t$ for $i = 0, \dots, n$, $\lambda_{-1}(t) := 0$, and $\lambda_{n+1}(t) := 1$. Indeed, we can write (1.6) as

$$(b_0^{n+1}(t), \dots, b_{n+1}^{n+1}(t)) = (b_0^n(t), \dots, b_n^n(t))\Lambda(t), \tag{1.7}$$

where $\Lambda(t)$ denotes the nonnegative stochastic bidiagonal matrix

$$\Lambda(t) = \begin{pmatrix} 1 - \lambda_0(t) & \lambda_0(t) & & \\ & \ddots & \ddots & \\ & & 1 - \lambda_n(t) & \lambda_n(t) \end{pmatrix}. \tag{1.8}$$

Then, starting with a Bézier curve $\gamma(t) = \sum_{i=0}^{n+1} Q_i b_i^{n+1}(t)$, (1.7) gives

$$\gamma(t) = (b_0^{n+1}, \dots, b_{n+1}^{n+1})(Q_0, \dots, Q_{n+1})^T = (b_0^n, \dots, b_n^n)(P_0(t), \dots, P_n(t))^T,$$

where

$$(P_0(t), \dots, P_n(t))^T := \Lambda(t)(Q_0, \dots, Q_{n+1})^T. \tag{1.9}$$

Equality (1.9) describes the first step of the de Casteljau algorithm for the evaluation of $\gamma(t)$.

Bernstein bases are totally positive on $[0, 1]$. In this paper we shall prove that similar properties hold for any totally positive basis of functions. Let us recall that a totally positive matrix is a matrix such that all of its minors are nonnegative. A totally positive system of functions defined on I is a system $(u_0, \dots, u_n)$ such that $(u_j(t_i))_{0 \le i,j \le n}$ is totally positive for all $t_0 < \cdots < t_n$ in I. A normalized totally positive (NTP) basis $(u_0, \dots, u_n)$ is a totally positive system of linearly independent functions such that $\sum_{i=0}^n u_i = 1$.

Given an NTP basis $(u_0^{n+1}, \dots, u_{n+1}^{n+1})$ on an interval I, and a nonnegative stochastic $(n+2) \times (n+1)$ matrix A of rank $n+1$, we shall consider the system of functions defined by

$$(u_0^n, \dots, u_n^n) := (u_0^{n+1}, \dots, u_{n+1}^{n+1})A. \tag{1.10}$$

Starting from a curve $\gamma(t) = \sum_{i=0}^{n} P_i u_i^n(t)$, $t \in I$, clearly (1.10) allows us to express it as $\gamma(t) = \sum_{i=0}^{n+1} Q_i u_i^{n+1}(t)$, where the points Q_i are defined by (1.5). In this paper we will derive from (1.10) the existence of nondecreasing functions $\lambda_0, \ldots, \lambda_n$ with values in $[0,1]$ such that

$$(u_0^{n+1}(t), \ldots, u_{n+1}^{n+1}(t)) = (u_0^n(t), \ldots, u_n^n(t))\Lambda(t), \quad t \in I. \tag{1.11}$$

The matrix $\Lambda(t)$ is defined from λ_i as in (1.8). Starting with a curve $\gamma(t) = \sum_{i=0}^{n+1} Q_i u_i^{n+1}(t)$, $t \in I$, we will be able to write it as $\gamma(t) = \sum_{i=0}^{n} P_i(t) u_i^n(t)$, where the points $P_0(t), \ldots, P_n(t)$ are again given by (1.9). On the other hand, we shall check that (1.10) implies that $(u_0^n, \ldots, u_n^n)$ is an NTP basis on I. It will therefore be possible to iterate this process. Doing so, we shall obtain a de Casteljau type algorithm for the evaluation of $\gamma(t)$.

Pottmann and Mazure in [5,6,7] developed generalizations (1.11) of the de Casteljau algorithm for Tchebycheffian curves. Here we show that these generalizations can be also obtained for any curve generated by an NTP basis.

We observe that for each value of t, the point $P_i^j(t)$ is a convex combination of two consecutive points, obtained in the previous step of the algorithm. Therefore, these algorithms can be seen as corner cutting algorithms for curve evaluation [4].

§2. Recurrence Relations for NTP Systems

The following proposition allows us to describe the generalization of formulae (1.6) to any NTP systems related by a matrix (1.3). First we need to show the following auxiliary result.

Lemma 2.1. *Let $(u_0^{n+1}, \ldots, u_{n+1}^{n+1})$ be an NTP basis of functions defined on I and*

$$(u_0^n, \ldots, u_n^n) := (u_0^{n+1}, \ldots, u_{n+1}^{n+1})A, \tag{2.1}$$

where $A \in \mathbb{R}^{(n+2)\times(n+1)}$ is of the form (1.3). Let $C_i := \{t \in I \mid u_i^n(t) \neq 0\}$. Then

(i) *$u_{i+1}^{n+1}(t)/u_i^n(t)$, $t \in C_i$, is a nondecreasing function,*

(ii) *$\alpha_{i+1} u_{i+1}^{n+1}(t)/u_i^n(t) \in [0,1]$, for all $t \in C_i$.*

Proof:

(i) Since A is bidiagonal, we can write

$$u_i^n(t) = (1-\alpha_i) u_i^{n+1}(t) + \alpha_{i+1} u_{i+1}^{n+1}(t), \quad t \in I, \qquad i = 0, \ldots, n. \tag{2.2}$$

Observe that, since A is nonnegative and $(u_0^{n+1}, \ldots, u_{n+1}^{n+1})$ is NTP, then $u_i^n(t) > 0$, for all $t \in C_i$. Moreover,

$$\begin{vmatrix} u_i^n(t) & u_{i+1}^{n+1}(t) \\ u_i^n(s) & u_{i+1}^{n+1}(s) \end{vmatrix} = (1-\alpha_i) \begin{vmatrix} u_i^{n+1}(t) & u_{i+1}^{n+1}(t) \\ u_i^{n+1}(s) & u_{i+1}^{n+1}(s) \end{vmatrix} \geq 0, \quad t < s, \tag{2.3}$$

because $(u_0^{n+1}, \ldots, u_{n+1}^{n+1})$ is totally positive. Formula (2.3) implies that u_{i+1}^{n+1}/u_i^n is nondecreasing in C_i.

(ii) Using (2.2), we can write

$$0 \le \frac{\alpha_{i+1}u_{i+1}^{n+1}(t)}{u_i^n(t)} = \frac{\alpha_{i+1}u_{i+1}^{n+1}(t)}{(1-\alpha_i)u_i^{n+1}(t)+\alpha_{i+1}u_{i+1}^{n+1}(t)} \le 1, \quad t \in C_i. \quad \square$$

The following proposition is devoted to showing that formula (1.11) holds for NTP bases.

Proposition 2.2. *Let $(u_0^{n+1},\dots,u_{n+1}^{n+1})$ and $(u_0^n,\dots,u_n^n)$ be two NTP bases of functions on I related by (2.1), where A is a matrix (1.3) (rank $A = n+1$). Let $C_i := \{t \in I \mid u_i^n(t) \neq 0\}$. Then the functions $\lambda_i : I \to \mathbb{R}$, $i = 0,\dots,n$, defined by*

$$\lambda_i(t) := \begin{cases} \alpha_{i+1} \inf \{u_{i+1}^{n+1}(s)/u_i^n(s) \mid s \in C_i\}, & \text{if } u_i^n(s) = 0, \forall s \le t, \\ \alpha_{i+1} \sup \{u_{i+1}^{n+1}(s)/u_i^n(s) \mid s \in C_i, s \le t\}, & \text{otherwise,} \end{cases} \tag{2.4}$$

are nondecreasing, and satisfy

$$0 \le \lambda_i(t) \le 1, \quad \forall t \in I, \qquad i = 0,\dots,n. \tag{2.5}$$

Furthermore, if we use definition (1.8), then (1.11) holds.

Proof: Since $(u_0^n,\dots,u_n^n)$ are linearly independent, then $C_i \neq \emptyset$ for all i. Therefore, by Lemma 2.1 (ii), we can define

$$\kappa_i := \alpha_{i+1} \inf \left\{ \frac{u_{i+1}^{n+1}(s)}{u_i^n(s)} \,\middle|\, s \in C_i \right\} \in [0,1].$$

If the condition $u_i^n(s) = 0$, $\forall s \le t$, does not hold, then the set $\{s \in C_i \mid s \le t\}$ is nonempty and by Lemma 2.1 (ii), we can define

$$\kappa_i \le \lambda_i(t) := \alpha_{i+1} \sup \left\{ \frac{u_{i+1}^{n+1}(s)}{u_i^n(s)} \,\middle|\, s \in C_i, s \le t \right\} \le 1.$$

We have seen that λ_i, $i = 0,\dots,n$, are well-defined and that (2.5) holds. In order to see that $\lambda_i(t)$ are nondecreasing, let us observe first that if $\{s \in C_i \mid s \le t_1\} = \emptyset$ and $\{s \in C_i \mid s \le t_2\} \neq \emptyset$, then t_1 must be less than t_2. Therefore, we only have to show that $\lambda_i(t_1) \le \lambda_i(t_2)$ only for all $t_1 < t_2$ such that there exists some $s < t_1$ with $u_i(s) \neq 0$. We observe that $\{s \in C_i \mid s \le t_1\} \subseteq \{s \in C_i \mid s \le t_2\}$. Therefore, by Lemma 2.1 (i),

$$\alpha_{i+1} \sup \left\{ \frac{u_{i+1}^{n+1}(s)}{u_i^n(s)} \,\middle|\, s \in C_i, s \le t_1 \right\} \le \alpha_{i+1} \sup \left\{ \frac{u_{i+1}^{n+1}(s)}{u_i^n(s)} \,\middle|\, s \in C_i, s \le t_2 \right\}.$$

We now establish the relation

$$\alpha_{i+1}u_{i+1}^{n+1}(t) = \lambda_i(t)u_i^n(t), \quad \forall t \in I, \quad i = 0,\dots,n. \tag{2.6}$$

If $u_i^n(t) = 0$, then by (2.2), $\alpha_{i+1}u_{i+1}^{n+1}(t) = 0$, and (2.6) trivially holds. Otherwise, we have $t \in \{s \in C_i \mid s \le t\}$, and by Lemma 2.1 (i),

$$\lambda_i(t) = \alpha_{i+1}u_{i+1}^{n+1}(t)/u_i^n(t),$$

so (2.6) is confirmed again.

Finally, using (2.6) and (2.2) we can write

$$\lambda_{i-1}(t)u_{i-1}^n(t) + (1-\lambda_i(t))u_i^n(t) = \alpha_i u_i^{n+1}(t) + u_i^n(t) - \alpha_{i+1}u_{i+1}^{n+1}(t) = u_i^{n+1}(t)$$

for all $t \in I$ and $i = 0,\dots,n$. $\square$

§3. A Generalization of the de Casteljau Algorithm for NTP Bases

Given an NTP basis $(u_0^n, \dots, u_n^n)$ of a space $\mathcal{U}^n$ of functions defined on I, we can obtain a sequence of NTP bases $(u_0^k, \dots, u_k^k)$ of $(k+1)$-dimensional subspaces $\mathcal{U}^k$ by the recurrence

$$(u_0^k(t), \dots, u_k^k(t)) := (u_0^{k+1}(t), \dots, u_{k+1}^{k+1}(t))A_{k+1}, \ k = n-1, n-2, \dots, 0, \quad (3.1)$$

where $A_{k+1} \in \mathbb{R}^{(k+2)\times(k+1)}$ is a matrix of type (1.3), $\operatorname{rank} A_{k+1} = k+1$.

In fact, since A_{k+1} are nonnegative bidiagonal matrices, it easily follows, using Theorem 2.3 of [1], that A_{k+1} is totally positive and, using the Cauchy-Binet formula (formula (1.23) of [1]), that the systems (3.1) are totally positive. Taking into account that $(u_0^{k+1}, \dots, u_{k+1}^{k+1})$ is normalized and A_{k+1} is stochastic, we derive that the systems (3.1) are also normalized. Furthermore, formula (3.1) relates two bases if and only if $\operatorname{rank} A_{k+1} = k+1$. Observe that $\operatorname{rank} A_{k+1} < k+1$ if and only if there exist $1 \le i < j \le k$ such that $\alpha_i^{k+1} = 1$ and $\alpha_j^{k+1} = 0$.

Let us observe that the subspaces $\mathcal{U}^k$ form a chain, that is,

$$\mathcal{U}^n \supset \mathcal{U}^{n-1} \supset \cdots \supset \mathcal{U}^1 \supset \mathcal{U}^0 = \operatorname{span}\{1\}.$$

Moreover, since (u_0^0) is an NTP basis of $\mathcal{U}^0$ then $u_0^0(t) = 1$, for all $t \in I$.

By Proposition 2.2, the bases of (3.1) are related by

$$(u_0^{k+1}(t), \dots, u_{k+1}^{k+1}(t)) = (u_0^k(t), \dots, u_k^k(t))\Lambda_{k+1}(t), \quad t \in I, \quad (3.2)$$

where $\Lambda_{k+1}(t)$ is a matrix of type (1.8). We shall denote by $\lambda_i^{k+1}(t)$ the $(i+1, i+2)$ entry of $\Lambda_{k+1}(t)$. The recurrences (3.1) and (3.2) give

$$(u_0^k(t), \dots, u_k^k(t)) = (u_0^n(t), \dots, u_n^n(t))A_n \cdots A_{k+2}A_{k+1}, \quad t \in I, \quad (3.3)$$

and

$$(u_0^k(t), \dots, u_k^k(t)) = \Lambda_1(t)\Lambda_2(t)\cdots\Lambda_k(t), \quad t \in I, \quad (3.4)$$

for $k = 0, \dots, n$, with the convention $A_n \cdots A_{k+1}$ equals the identity matrix when $k = n$ and $\Lambda_1(t)\cdots\Lambda_k(t)$ equals the scalar constant 1 when $k = 0$.

Formulae (3.4) can be interpreted as a factorization of the NTP system $(u_0^k, \dots, u_k^k)$ as a product of bidiagonal stochastic matrices of functions.

Let us summarize all the conclusions in the following theorem.

Theorem 3.1. *Let $(u_0^n, \dots, u_n^n)$ be an NTP basis of functions defined on I. Let $A_k \in \mathbb{R}^{(k+1)\times k}$, $k = 1, \dots, n$, be matrices (3.1) of maximal rank. Define NTP systems $(u_0^k, \dots, u_k^k)$, $k = 0, \dots, n-1$, by (3.1) (or equivalently by (3.3)). Then there exist matrices $\Lambda_k(t)$ of type (1.8) whose $(i+1, i+2)$ entry $\lambda_i^k(t)$ is nondecreasing on I and with values in $[0,1]$, $k = 1, \dots, n$, such that (3.2) and (3.4) hold. In particular,*

$$(u_0^n(t), \dots, u_n^n(t)) = \Lambda_1(t)\cdots\Lambda_n(t), \quad \forall t \in I. \quad (3.5)$$

Moreover, for any control polygon $P_0 \cdots P_n$ consider the following generalization of the de Casteljau algorithm:

$$\begin{aligned}
&\textbf{for } j = 0, 1, \ldots, n \\
&\qquad P_j^n(t) := P_j \\
&\textbf{for } i = n-1, \ldots, 1, 0 \\
&\qquad \textbf{for } j = 0, 1, \ldots, i \\
&\qquad\qquad P_j^i(t) := (1 - \lambda_j^{i+1}(t)) P_j^{i+1}(t) + \lambda_j^{i+1}(t) P_{j+1}^{i+1}(t)
\end{aligned}$$

At each step we have

$$\gamma(t) = \sum_{j=0}^{i} P_j^i(t) u_j^i(t), \quad t \in I, \qquad i = 0, \ldots, n. \tag{3.6}$$

In particular $\gamma(t) = P_0^0(t)$ for all $t \in I$, that is, this generalized de Casteljau algorithm reconstructs the curve from its control polygon.

Proof: The existence of the matrices $\Lambda_k(t)$ of type (1.8), satisfying (3.1) and (3.2) follows from Proposition 2.2. From the algorithm we see that

$$(P_0^i(t), \ldots, P_i^i(t))^T = \Lambda_{i+1}(t)(P_0^{i+1}(t), \ldots, P_{i+1}^{i+1}(t))^T,$$

and by (3.5) we can write

$$\begin{aligned}
\gamma(t) =& (u_0^n(t), \ldots, u_n^n(t))(P_0, \ldots, P_n)^T = \\
& \Lambda_1(t) \cdots \Lambda_k(t) \Lambda_{k+1}(t) \cdots \Lambda_n(t)(P_0, \ldots, P_n)^T = \\
& (u_0^k(t), \ldots, u_k^k(t))(P_0^k(t), \ldots, P_k^k(t))^T. \quad \square
\end{aligned}$$

Example 3.2. When applying Proposition 2.2 to (1.1) or (1.2), the functions that we obtain are $\lambda_i(t) = t$, $i = 0, \ldots, n$. Hence we obtain (1.7), and the corresponding algorithm described in Theorem 3.1 is just the classical de Casteljau algorithm for polynomials. Of course, any other choice of a sequence $(A_k), k = 1, \ldots, n$, of nonnegative stochastic matrices of maximal rank could lead to another de Casteljau type algorithm. For instance, if we consider the Bernstein basis (b_0^2, b_1^2, b_2^2) of degree 2, the matrix

$$A_2 := \begin{pmatrix} 1 & 0 \\ 1/3 & 2/3 \\ 0 & 1 \end{pmatrix}$$

defines a NTP basis $((1-t)(3-t)/3, t(4-t)/3)$ on $[0, 1]$. This system generates a subspace of quadratic functions, different from the polynomials of degree less than or equal to 1. Furthermore, the functional matrices obtained by applying Proposition 2.2

$$\Lambda_1(t) = \begin{pmatrix} \frac{(1-t)(3-t)}{3} & \frac{t(4-t)}{3} \end{pmatrix}, \quad \Lambda_2(t) = \begin{pmatrix} \frac{3(1-t)}{3-t} & \frac{2t}{3-t} & 0 \\ 0 & \frac{4(1-t)}{4-t} & \frac{3t}{4-t} \end{pmatrix},$$

lead to a corner cutting algorithm different from the classical de Casteljau algorithm.

In [2], it was shown that, in any space with an NTP basis, there exists a particular NTP basis called the normalized B-basis which has the optimal shape preserving properties among all NTP bases of the space. In Theorem 4.3 of [3] it was shown that if $(u_0^{n+1}, \dots, u_{n+1}^{n+1})$ is a normalized B-basis of an $(n+2)$-dimensional space and $(u_0^n, \dots, u_n^n)$ is a B-basis of an $(n+1)$-dimensional subspace, then there exists a matrix A (1.3) such that

$$(u_0^n, \dots, u_n^n) = (u_0^{n+1}, \dots, u_{n+1}^{n+1})A.$$

Thus, B-bases provide good examples of when Theorem 3.1 can be applied. In the case of polynomial spline spaces (see [2]), the normalized B-basis is precisely the B-spline basis.

Let $\mathcal{T} = \{t_0 = \cdots = t_{k-1} < t_k \le \cdots \le t_n < t_{n+1} = \cdots = t_{n+k}\}$, $t_i < t_{i+k}$, for all i, be an extended knot sequence and

$$N_{i,\mathcal{T}}^k(t) := (t_{i+k} - t_i)[t_i, \dots, t_{i+k}](\cdot - t)_+^{k-1}, \quad t \in [t_0, t_{n+1}],\ i = 0, \dots, n,$$

the associated B-spline basis of the space $\mathcal{S}_{\mathcal{T}}^k$. Let us insert a knot τ in $\mathcal{T}$ such that $t_j \le \tau < t_{j+1}$ (if $\tau = t_j$ then the multiplicity of t_j must be less than k) and define a new sequence of knots $\hat{\mathcal{T}}$

$$\hat{t}_i := \begin{cases} t_i, & 0 \le i \le j, \\ \tau, & i = j+1, \\ t_{i-1}, & j+2 \le i \le n+k+1. \end{cases}$$

The normalized B-bases of $\mathcal{S}_{\mathcal{T}}^k$ and $\mathcal{S}_{\hat{\mathcal{T}}}^k$ are related by a matrix (1.3) with

$$\alpha_i := \begin{cases} 0, & 0 \le i \le j-k+1, \\ (t_{i+k-1} - \tau)/(t_{i+k-1} - t_i), & j-k+2 \le i \le j, \\ 1, & j+1 \le i \le n. \end{cases}$$

Applying Proposition 2.2 to both B-spline bases, a relation (1.11) is obtained. In order to obtain a generalized de Casteljau algorithm, we first remove succesively all interior knots until we arrive at the Bernstein basis. Then we can continue with the steps of an evaluation algorithm for polynomials (e.g., the de Casteljau algorithm). We illustrate this procedure with a simple example:

Example 3.3. Take $\hat{\mathcal{T}} := (0, 0, 1/2, 1, 1)$, $\mathcal{T} := (0, 0, 1, 1)$. The associated B-spline bases are related by a matrix (1.3):

$$\left(N_{0,\mathcal{T}}^2(t), N_{1,\mathcal{T}}^2(t)\right) = \left(N_{0,\hat{\mathcal{T}}}^2(t), N_{1,\hat{\mathcal{T}}}^2(t), N_{2,\hat{\mathcal{T}}}^2(t)\right) \begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \\ 0 & 1 \end{pmatrix}, \quad t \in [0, 1].$$

Using Proposition 2.2, we obtain that these bases are also related by (1.11) with a matrix (1.8), where

$$\lambda_0(t) = \min(1, t/(1-t)), \quad \lambda_1(t) = \max(0, (2t-1)/t), \quad t \in [0, 1].$$

The evaluation algorithm for $\gamma(t) := \sum_{i=0}^{2} P_i N_{i,\hat{T}}^2(t)$ can be described as follows. First compute

$$P_0^1(t) := (1-\lambda_0(t))P_0 + \lambda_0(t)P_1, \quad P_1^1(t) := (1-\lambda_1(t))P_1 + \lambda_1(t)P_2,$$

and then $\gamma(t) = (1-t)P_0^1(t) + tP_1^1(t)$. Note that the last step of the algorithm corresponds to the de Casteljau algorithm. Of course, this algorithm is different from the classical de Boor-Cox algorithm for evaluation of B-splines.

Acknowledgments. This research has been partially supported by Research Grant DGES PB96–0730. We thank the referee for his valuable suggestions.

References

1. Ando, T., Totally positive matrices, Linear Algebra Appl. **90** (1987), 165–219.
2. Carnicer, J. M. and J. M. Peña, Totally positive bases for shape preserving curve design and optimality of B-splines, Comput. Aided Geom. Design **11** (1994), 635–656.
3. Carnicer, J. M. and J. M. Peña, Characterizations of the optimal Descartes' rules of signs, Math. Nachrichten **189** (1998), 33–48.
4. Goodman, T. N. T. and C. A. Michelli, Corner cutting algorithms for the Bézier representation of free form curves, Linear Algebra Appl. **99** (1988), 225–252.
5. Mazure, M. L., Blossoming of Chebysheff splines, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1995, 355–364.
6. Mazure, M. L. and H. Pottman, Tchebycheff curves, in *Total Positivity and its Applications*, M. Gasca and C. A. Micchelli (eds.), Kluwer Academic, Dordrecht, 1996, 187–218.
7. Pottman, H., The geometry of Tchebycheffian splines, Comput. Aided Geom. Design **10** (1993), 181–210.

J. M. Carnicer
Departamento de Matemática Aplicada
Universidad de Zaragoza
50009 Zaragoza
Spain
carnicer@posta.unizar.es

E. Mainar
Departamento de Matemática Aplicada
Universidad de Zaragoza
50009 Zaragoza
Spain
esme@posta.unizar.es

Intersections et Convergence

Paul de Faget de Casteljau

Abstract. C'est en 1958, que j'ai rejoint l'équipe de fraisage numérique, CAO de la préhistoire, animée par Monsieur de la Boixière : Farfelus pour les uns, fous pour d'autres, personne en dehors de notre chef n'aurait osé parier un seul kopek sur l'avenir de cette technique. Les problèmes à surmonter étaient aussi nombreux que fort ardus : Par exemple, celui d'intersections, ou racines. Nos ordinateurs se bloquaient. De toute urgence, il fallait mettre sur pied une méthode efficace, en harmonie avec les exigences de ce nouvel outil. Après tant d'essais d'autant plus volumineux, que longtemps infructueux, comme tout parait simple, quand on présente le joyau final! Mieux encore, une idée directrice, qui comblera d'aise les férus de fractions continues, nous y conduit en toute logique.

§1. Un Peu d'Histoire

Pour démarrer l'étude des formes de C.A.O., quelques instants de réflexion pouvaient suffire, pour se convaincre que, pour concilier les impératifs du dessin à ceux d'une production de calculs, la seule solution acceptable restait l'utilisation en géométrie affine des formes paramétriques polynomiales à travers l'algorithme devenu célèbre. La nécessité d'obtenir une solution, unique pour tout paramètre, interdisait des calculs aventureux. Même si le calcul de fonctions du genre arc $\cos(x)$, au voisinage de $x = 0$, aurait gagné du point de vue logique, de l'utilisation de formes $\sqrt{x}$, déjà multiforme, les mathématiques en sont restées aux séries entières! (rarement, les fractions rationnelles, fractions continues...).

Il n'en restait pas moins à traiter de nombreux problèmes, qui, chacun, pouvait me donner l'impression d'avoir à passer le concours de l'agrégation, me créant des cauchemars nocturnes, tout en regrettant amèrement le bon et solide poste de professeur qui aurait pu être le mien.

"M'sieur, ça ne marche pas!", phrase rituelle, qui m'annonçait encore une nouvelle catastrophe, à cette époque héroïque, pour ne pas dire préhistorique. Cela m'arrachait à mes méditations et rêveries où je me figurais dans une sorte de paradis terrestre, où les mêmes visiteurs me félicitaient de ma dernière

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 9–15.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

trouvaille d'un "Ce n'est vraiment pas mal, votre truc!", phrase toujours restée dans le virtuel de ma pensée.

Parmi ces innombrables problèmes, qui me ramenaient brutalement au sens des réalités, il y eut le problème des intersections. Là encore, pas question d'adopter une attitude universitaire, autour d'une savante discussion sur l'existence et la nature d'éventualités favorables. Il fallait "produire" une racine et une seule, et de surcroît, de valeur acceptable. Cette fois, l'unicité devenait difficile à garantir, surtout au voisinage de racines multiples, voisines ou confondues. Aucune des tentatives de tenir compte des termes non linéaires (Whittaker, etc...) pour améliorer la méthode de Newton ne s'est vraiment imposée. Et les exploitations en ordinateurs atteignaient la limite du quart d'heure qui, par précaution, en limitait la durée! Et de plus, il fallait agir vite, très vite...pour reprendre un problème très classique à zéro, en poussant notre "cri de guerre" : "Mais, qu'alors y faire?" (jeu de mots: allusion à un calorifère).

§2. Exposé du Problème

On veut calculer la valeur de x, qui vérifie l'équation

$$P(x) = 0,$$

où $P(x)$ est un polynôme (ou un développement limité, en série de Taylor, d'une fonction $f(x)$...).

Ce problème se pose aussi à plusieurs inconnues; ainsi, pour $n = 3$, calcul de x, y, z pour que

$$\begin{cases} P(x,y,z) &= 0, \\ Q(x,y,z) &= 0, \\ R(x,y,z) &= 0. \end{cases}$$

Pour le cas $n = 2$, j'ai présenté au congrès de Schloß Dagstuhl (Saarland 1996) une fort jolie méthode, habile manipulation de géométrie projective, très efficace; malheureusement, tous mes efforts de l'étendre à d'autres valeurs de n ont échoué. J'ajouterai à l'adresse des chercheurs, qui auraient la témérité d'étudier le cas $n = 3$, que j'avais bien cerné la difficulté qui rendait ce cas impossible.

Les seules équations qui conduisent à une solution unique, sont du premier degré. Il nous reste la possibilité de tenter une identification autour de $x = 0$ du polynôme $P(x)$ à une fraction rationnelle dont le numérateur soit linéaire.

$$n = 1 \qquad P(x) \sim \frac{a + bx}{1 + dx + ex^2 + \ldots}$$

$$n = 3 \qquad P(x,y,z) \sim \frac{a + bx + cy + dz}{1 + \sum h_p x^\alpha y^\beta z^\gamma}$$

On en déduit une suite d'approximations de la racine, chacune utilisant les précédentes, de façon optimale. Cela peut se prouver par l'analyse, et

le vérifier sur des exemples concrets : elle est la meilleure, comparée à toute manipulation du même type.

L'identification brutale, terme à terme, conduit à

$$A + Bx + Cx^2 \sim \frac{AB + (B^2 - AC)x}{B - Cx + \dots}$$

$$A + Bx + Cx^2 + Dx^3 \sim \frac{A(B^2 - AC) + [A(AD - BC) + B(B^2 - AC)]x}{B^2 - AC + (AD - BC)x + (C^2 - BD)x^2 + \dots}$$

$$A + Bx + Cx^2 + Dx^3 + Ex^4 \sim \frac{\text{num}}{\text{den}},$$

$$\begin{aligned}
\text{num} = {} & A[A(AD - BC) + B(B^2 - AC)] \\
& + [A^2(BD - AE) + AB(AD - BC) + (B^2 - AC)^2]x \\
\text{den} = {} & [A(AD - BC) + B(B^2 - AC)] \\
& + [A(BD - AE) - C(B^2 - AC)]x \\
& + [B(AE - BD) - C(AD - BC)]x^2 \\
& - [A(D^2 - CE) + B(BF - CD) + C(C^2 - BD)]x^3 + \dots
\end{aligned}$$

Seuls les numérateurs nous intéressent :

$$\begin{vmatrix}
B & A & 0 & 0 & 0 & \dots \\
C & B & A & 0 & 0 & \dots \\
D & C & B & A & 0 & \dots \\
E & D & C & B & A & \dots \\
F & E & D & C & B & \dots \\
\dots & \dots & & & & \\
\dots & \dots & & & &
\end{vmatrix}$$

Si on appelle U_i la succession des déterminants utilisant les i premières lignes et colonnes de la matrice ci-dessus, en posant $U_0 = 1$, on obtient la séquence d'approximations

$$x_i = -\frac{A\,U_{i+1}}{U_i},$$

$x_1 = -\frac{A}{B}$ (Newton), x_2 coïncide avec Whittaker, mais non x_3 et au delà.
Mais il y a mieux à faire !

§3. Algorithme

Une étude complète de cette question prouve que l'on obtient le même résultat en utilisant un algorithme qui rappelle le schéma de Hörner.

Il s'agit de calculer la racine x, en partant de la valeur $x = 0$ d'une série entière type MacLaurin écrite sous la forme

$$a_0 = a_1 x + a_2 x^2 + \dots + a_n x^n + \dots$$

On écrit alors l'algorithme sous la forme :

$a_0 = a_1 N$	donne $N \sim$ Newton	
$a_0 = (a_1 + a_2 N)P$	" $P \sim$ Whittaker	1
$a_0 = [a_1 + (a_2 + a_3 N)P]Q$	" $Q \sim$ Fractions continues	2^2
$a_0 = \{a_1 + [a_2 + (a_3 + a_4 N)P]Q\}R$	" R	5^3
		130^4
... etc ...	" $S, T, \ldots$	

On a indiqué à droite le nombre (*) de possibilités d'utilisation des approximations antérieures, dans un ordre différent, ou avec répétition. Mes études, à l'époque, m'ont conduit à prouver qu'aucune ne fait mieux, avec les hypothèses adoptées (le nombre 130 est la somme des trois précédents, pour tenir compte de toutes les approximations nouvelles).

Cette méthode est d'autant plus efficace que la série à inverser est plus convergente ; cela se vérifie pour e^x, $\sin(x)$, $\mathrm{sh}(x)$, pour calculer $\ln(x)$, Arc $\sin(x)$, Arg $\mathrm{sh}(x)$.

On va, à titre d'exemple, calculer le logarithme népérien de 2, là où la série $\ln(1+x)$, au bord de son cercle de convergence, est particulièrement désespérante.

Il convient de n'y voir qu'un exemple, et non une invitation à faire le calcul des logarithmes de cette façon-là. Notons encore tout l'intérêt d'inverser les lignes trigonométriques par l'utilisation *simultanée* du sinus et du cosinus, ou encore pour un triangle rectangle les cotés S, C, H du sinus, du cosinus et l'hypoténuse.

Pour les lecteurs intéressés par cette question, on peut démontrer deux séquences de réduites de fractions continues, coïncidant une fois sur deux, ou encadrant par excès et par défaut, la valeur exacte de la fonction : l'une de ces séquences se rattache au développement de Arc $\mathrm{tg}(\varphi/2) = t$ avec $S = 2t$, $C = 1 - t^2$, $H = 1 + t^2$, l'autre est originale. A l'époque où le calcul des fonctions n'était pas encore très au point, au niveau des ordinateurs, les formules simplifiées de Arc sin et Arc cos nous ont rendu un grand service. Ainsi, $\varphi \# S.\frac{14+C}{9+6C}$ qui n'est pas ridicule à $S = 1$, $C = 0$: $\varphi = \frac{\pi}{2} \sim \frac{14}{9}$. Mais cela n'est pas le sujet de cet exposé.

Exemple numérique :

Appliquons cet algorithme au calcul de x, défini par :

$$2 = e^x \sim 1 + \frac{x}{1}(1 + \frac{x}{2}(1 + \frac{x}{3}(1 + \ldots(\ldots(1 + \frac{x}{n}) \quad \ldots \quad)\ldots)))$$

On trouve ainsi :

(*) on pourrait encore considérer les possibilités obtenues, en prenant les formes polaires des N, P, Q. Ainsi, à la seconde ligne, au lieu de $a_1 P$, on écrirait $a_1 \frac{N+P}{2}$, etc...

$$
\begin{array}{lcrl}
N & = & 1 = & 1.00 \\
P & = & \frac{2}{3} = & 0.\underline{6}66\ 6 \\
Q & = & \frac{9}{13} = & 0.\underline{692}\ 30 \\
R & = & \frac{52}{75} = & 0.\underline{693}\ 333 \\
S & = & \frac{375}{541} = & 0.\underline{693}\ \underline{1}608 \\
T & = & \frac{3\,246}{4\,683} = & 0.\underline{693}\ \underline{145}\ 41 \\
U & = & \frac{32\,781}{47\,293} = & 0.\underline{693}\ \underline{146}\ 977 \\
V & = & \frac{378\,344}{545\,835} = & 0.\underline{693}\ \underline{147}\ \underline{196}\ 49 \\
W & = & \frac{4\,912\,515}{7\,087\,261} = & 0.\underline{693}\ \underline{147}\ \underline{183}\ 337 \\
X & = & \frac{70\,872\,610}{102\,247\,563} = & 0.\underline{693}\ \underline{147}\ \underline{180}\ 436 \\
 & & & \ldots\ldots\ldots\ldots \\
 & & & \ldots\ldots\ldots\ldots \\
 & & & 0.693\ 147\ 180\ 559\ 945\ 309
\end{array}
$$

soit environ un gain d'une décimale à chaque itération, ce qui reste plus qu'honorable comparé avec les séries traditionnelles $\ln(1+u)$ avec $u=1$ ou $\ln(\frac{1+u}{1-u})$ avec $u=\frac{1}{3}$, bien plus convergente.

§4. Promenade au Pays du Logarithme

Il est toujours passionnant de reprendre à son compte l'étude des fonctions élémentaires, qui conduisent à des développements d'une telle richesse, qu'ils paraissent inépuisables. Ainsi en est-il avec le logarithme, l'étude qui suit, et qui n'est qu'un tout petit aperçu des propriétés que révèle l'analyse, sans parler des propriétés dévoilées par l'arithmétique.

L'étude précédente, reprise de façon littérale, founit la succession d'approximations

$(x-1)$; $2\frac{(x-1}{(x+1)}$; $3\frac{(x-1)(x+1)}{x^2+4x+1}$; $4\frac{(x-1)(x^2+4x+1)}{(x+1)(x^2+10x+1)}$; $5\frac{(x-1)(x+1)(x^2+10x+1)}{x^4+26x^3+66x^2+26x+1}$; ...

On constate déjà plusieurs règles empiriques. Sans insister, posons :

si $n=2p$: $\quad n\frac{(x-1)}{(x+1)}\frac{P_{2n-2}}{Q_{2n-2}}$, $\qquad$ si $n=2p+1$: $\quad n\,(x-1)(x+1)\frac{Q_{2n-2}}{P_{2n}}$.

On peut encore réduire l'écriture de ces polynômes reconnus symétriques, en posant $x+\frac{1}{x}-2=2t$ et si $k_q=1$ pour les polynômes P et $k_q=q$ pour les polynômes Q :

Q ou $P=1$

P_1 ou $Q_1=t+3k_2$

P_2 ou $Q_2=t^2+15k_2t+30k_3$

P_3 ou $Q_3=t^3+63k_2t^2+420k_3t+630k_4$

P_4 ou $Q_4=t^4+255k_2t^3+4410k_3t^2+18900k_4t+22680k_5$

.....

On peut encore comparer ces fonctions avec les réduites du développement de $\ln(x)$ en fractions continues, très inhabituelles, parce qu'elles sont centrées autour de $x=1$ au lieu du zéro habituel. Elles sont très faciles à obtenir,

par intégration de formules $Q(x)\ln(x) - P(x) = k(x-1)^n$, alternées avec des changements de x en $1/x$, qui change le signe de $\ln(x)$.

On remarquera les carrés des coefficients du binôme, dans les expressions des dénominateurs de rang pair (cela se démontre).

$$(x-1) \;;\; \frac{2(x-1)}{x+1} \;;\; \frac{x^2+4x-5}{4x+2} \;;\; \frac{3(x^2-1)}{x^2+4x+1} \;;\; \frac{x^3+18x^2-9x-10}{9x^2+18x+3} \;;$$
$$\frac{11x^3+27x^2-27x-11}{3(x^3+9x^2+9x+1)} \;;\; \frac{3x^4+128x^3+108x^2-192x-47}{12(4x^3+18x^2+12x+1)} \;;\; \frac{25x^4+160x^3-160x-25}{6(x^4+16x^3+36x^2+16x+1)} \cdots$$

Si l'on pose $D_n = \frac{(x-1)^n}{\Sigma (C_n^p)^2 x^p}$ et en utilisant la valeur des réduites paires, on peut même arriver à établir cette formule particulièrement brillante :

$$\ln(x) \sim 2\,[\,\frac{D_1}{1} + \frac{D_1 D_2}{2} + \frac{D_2 D_3}{3} + \frac{D_3 D_4}{4} + \cdots + \frac{D_{n-1} D_n}{n} + \cdots\,].$$

Remarque. L'étude arithmétique, à laquelle j'ai fait allusion, obéit à une démarche opposée à celle de la table des nombres premiers. On ne retient des nombres entiers que ceux dont les facteurs premiers ne sont que les n premiers nombres de cette table des nombres premiers. Les inégalités sur leurs logarithmes qui en découlent fournissent une série de solutions aux problèmes de Dirichlet-Hermite. On trouve ainsi une explication rationnelle aux gammes musicales... Voir à ce sujet les derniers chapitres de mon livre "Quaternions".

Ces considérations s'appliquent aussi à des fonctions. Ainsi trouver les meilleures solutions polynomiales à un degré donné, sensiblement proportionnelle à un arc, et à leurs sinus et cosinus. On peut y rajouter la tangente, etc...

§4. Cas de Plusieurs Variables

Le regard de l'analyse

Le problème est trop vaste pour être exposé en quelques lignes. Nous renvoyons aux ouvrages spécialisés, pour constater qu'il ne suffit pas de se donner p points, pour définir la valeur de p coefficients d'une interpolation à plusieurs paramètres. Il intervient alors la notion de silhouette du polynôme associé à une grille de points donnés. Les silhouettes les plus considérées, à deux variables, peuvent être ainsi triangulaires $\sum x^\alpha y^\beta$ avec $\alpha + \beta \leq n$, ou encore rectangulaires si $\alpha \leq n$, $\beta \leq q$ et carrées si $n = q$. Paradoxalement, c'est avec les grilles les plus régulières qu'on rencontre le plus de contraintes à respecter.

C'est en fonction de la notion de sihouette que l'on imposera un certain ordre de classement aux termes d'un polynôme à plusieurs variables. Pour déterminer une fraction rationnelle à numérateur linéaire au voisinage d'un point, on peut imaginer l'écriture d'un déterminant dont la première ligne s'écrit :

$$1, \quad x, \quad y, \quad f, \quad xf, \quad yf, \quad x^2 f, \quad xyf, \quad y^2 f, \ldots$$

la deuxième ligne exprime que f passe par le point X, Y, F (déterminant nul, si deux lignes égales) ensuite, un point voisin en X (puis en Y) et par différence avec cette seconde ligne s'exprimera à partir des dérivées partielles en X (puis en Y). On poursuivra les lignes suivantes avec les dérivées d'ordre supérieur.

Mais voilà ! On se heurte à un mur: le problème est indéterminé.

"Qu'alors y faire" ? Faut-il renoncer ? Le problème exige une solution, brillante, ou batarde, peu importe, à condition d'en trouver UNE ! Démarche industrielle, et non universitaire.

Alors, miracle ! Quelle que soit la façon dont on lève l'indétermination, par un choix arbitraire d'un terme de chaque degré du dénominateur, on retrouve le même numérateur; or c'est de lui et de lui seul qu'on a besoin, et tout rentre dans l'ordre.

Ici, honnêtement, je dois avouer, que je n'ai développé ce calcul, assez fastidieux, que dans le seul cas de deux variables.

Retour à l'algorithme

Il est plus rapide de faire un raisonnement, que d'aucuns jugeront fort acrobatique, mais parfaitement correct. Dans chaque équation, on suppose le problème résolu, pour toutes les variables, excepté une, d'abord x (puis $y, z, \ldots$). On applique alors notre algorithme au calcul de la première approximation de Newton donc N_x (puis N_y, N_z). On résoud le système, ce qui donne N_x, N_y, N_z.

On utilise ces valeurs approchées, comme on sait le faire pour déterminer P_x (puis P_y, P_z) et par suite P_x, P_y, P_z. Une troisième étape permettra le calcul de Q_x, Q_y, Q_z, etc ...

La meilleure silhouette qui convienne est donc carrée ($\alpha \leq n$, $\beta \leq n$, $\gamma \leq n$), ou complétée par des 0.

Remarque: Quel que soit le nombre de variables, l'optimisation du calcul montre que l'idéal se situe entre la seconde itération P_x, P_y, P_z, et la troisième Q_x, Q_y, Q_z. Après il est préférable de revenir à un développement de Taylor centré sur ce nouveau point, sauf cas de singularité extraordinaire, qu'en principe on ne doit pas rencontrer dans la pratique industrielle.

Voilà, en raccourci, l'exposé de ce problème très utile. Son étude approfondie pourrait faire l'objet d'une thèse complète. La discussion semble ici escamotée: ce n'est pas tout à fait vrai; car si une discussion s'impose, c'est que, dans la pratique industrielle, il n'y a plus de solution pratique. Par exemple, on ne cherche jamais l'intersection d'une forme avec une verticale, qui coupe la forme en trois points, intérieurs au carreau considéré. Comment l'usinerait-on ? Comment l'emboutir ?

Paul de Faget de Casteljau
4 Avenue du Commerce
78000 Versailles, France

Implicitization Matrices in the Style of Sylvester with the Order of Bézout

Eng-Wee Chionh, Ming Zhang, and Ronald Goldman

Abstract. Resultants are the standard tool used to compute the implicit equation of a rational curve or surface. Here we present a new way to compute the implicit equation of a rational curve by taking the determinant of a matrix having the style of the Sylvester resultant but the size of the Bézout resultant. Thus the new method has the advantages of both resultant schemes, representing the implicit equation as the determinant of a matrix with simple linear entries and lots of zeros just like the Sylvester resultant, but with the same small size as the Bézout resultant.

§1. Implicitization and Resultants

In Computer Aided Geometric Design (CAGD), curves and surfaces have two standard representations: parametric and implicit. The parametric representation is convenient for rendering curves and surfaces, whereas the implicit representation is useful for checking whether or not a point lies on a curve or surface. In the ideal situation, both representations are available. Given the parametric form of a curve or surface, one basic problem in CAGD is implicitization — that is, to find the implicit representation. Resultants are an effective tool for solving this problem for rational curves and surfaces [4,5].

Resultants are polynomial expressions in the coefficients of a set of polynomials; the vanishing of these expressions signals that the set of polynomials have a common root. For two univariate polynomials, there are two standard resultant formulations: Sylvester's resultant and Bézout's resultant. Given two degree n polynomials

$$f = \sum_{i=0}^{n} a_i t^i, \qquad g = \sum_{i=0}^{n} b_i t^i,$$

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 17–26.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

the Sylvester resultant is the determinant of the $2n \times 2n$ matrix

$$Syl(f,g) = \begin{bmatrix} a_0 & b_0 & & & & & \\ a_1 & b_1 & a_0 & b_0 & & & \\ \vdots & \vdots & a_1 & b_1 & \ddots & & \\ a_{n-1} & b_{n-1} & \vdots & \vdots & \ddots & a_0 & b_0 \\ a_n & b_n & a_{n-1} & b_{n-1} & \ddots & a_1 & b_1 \\ & & a_n & b_n & \ddots & \vdots & \vdots \\ & & & & \ddots & a_{n-1} & b_{n-1} \\ & & & & & a_n & b_n \end{bmatrix}.$$

Thus, the Sylvester resultant is just the determinant of the coefficient matrix of the polynomials $f, g, \cdots, t^{n-1}f, t^{n-1}g$ [6,9]. The Bézout resultant of f and g is the determinant of the $n \times n$ coefficient matrix $Bez(f,g)$, where $Bez(f,g)$ is defined by

$$\frac{\begin{vmatrix} f(t) & g(t) \\ f(\alpha) & g(\alpha) \end{vmatrix}}{\alpha - t} = [1 \quad \cdots \quad t^{n-1}] \cdot Bez(f,g) \cdot [1 \quad \cdots \quad \alpha^{n-1}]^T.$$

Explicit entry formulas for the Bézout resultant and fast computational algorithms for these entries can be found in [4,1].

The Sylvester and Bézout resultants each have certain advantages and disadvantages. The Sylvester resultant is sparse and all the nonzero entries of the Sylvester resultant come directly from the coefficients of f or g. The entries of the Bézout resultant are more complicated. However, to calculate the Sylvester resultant, a large determinant has to be computed, whereas the Bézout resultant matrix is much more compact.

To see why resultants arise naturally in implicitization, consider a rational curve

$$X = \frac{x(t)}{w(t)}, \qquad Y = \frac{y(t)}{w(t)}, \tag{1}$$

where $x(t), y(t), w(t)$ are polynomials. To obtain the implicit representation $F(X,Y) = 0$ for curve (1), introduce two auxiliary polynomials (in t)

$$X \cdot w(t) - x(t), \qquad Y \cdot w(t) - y(t). \tag{2}$$

By definition, the resultant of these two polynomials vanishes if and only if they have a common root, i.e. if and only if the point (X,Y) satisfies the two equations

$$X \cdot w(t) - x(t) = 0, \qquad Y \cdot w(t) - y(t) = 0,$$

for some value of t. Thus, (X,Y) makes the resultant of $X \cdot w(t) - x(t)$, $Y \cdot w(t) - y(t)$ vanish if and only if (X,Y) is on curve (1). So setting the resultant to zero yields the implicit equation of the parametric curve.

But which form of the resultant should one use? The Sylvester resultant has simple linear entries and lots of zeros, but to calculate the Sylvester resultant a large determinant has to be computed. The Bézout resultant has a more compact form, but the entries are much more complicated than the entries of the Sylvester resultant. Here we present a new way to compute the implicit equation of a rational curve by taking the determinant of a matrix having the style of the Sylvester resultant but the size of the Bézout resultant. Thus the new method has the advantages of both resultant schemes. That is, the new approach represents the implicit equation as the determinant of a matrix with simple linear entries and lots of zeros just like the Sylvester resultant, but the matrix has the same small size as the Bézout resultant.

Surfaces are beyond the scope of this work, but we hope to develop similar techniques for rational surfaces in a future paper [2].

§2. Implicitization from Moving Lines

In this section, we consider first rational curves of even degree. We begin by reviewing the concept of a moving line that follows a rational curve [7,8], and we show that there are always at least two moving lines of degree m that follow a rational curve of degree $2m$. The $m \times m$ Bézout determinant of these two moving lines has been used by previous authors to establish the efficacy of implicitization by the method of moving conics [3,8]. Here we prove that the $2m \times 2m$ Sylvester determinant of these two moving lines is an implicit expression for the rational curve if and only if there are no moving lines of degree $< m$ that follow the curve. This construction generates an implicitization matrix in the style of Sylvester with the order of Bézout. At the end of this section, we develop similar results for rational curves of odd degree.

2.1 Even degree rational curves

A rational curve of degree $2m$ can be written as $(x(t) : y(t) : w(t))$, where

$$x(t) = \sum_{i=0}^{2m} a_i t^i, \qquad y(t) = \sum_{i=0}^{2m} b_i t^i, \qquad w(t) = \sum_{i=0}^{2m} c_i t^i \tag{3}$$

and $\gcd(x(t), y(t), w(t)) = 1$. We can think of a rational curve as the track of a moving point.

Analogously, a moving line of degree d is defined by an implicit equation of the form

$$(A_0 x + B_0 y + C_0 w) + \cdots + (A_d x + B_d y + C_d w)t^d = 0, \tag{4}$$

where the coefficients $A_0, B_0, C_0, \cdots, A_d, B_d, C_d$ are constants. We say that the moving line (4) follows the rational curve (3) if and only if

$$(A_0 x(t) + B_0 y(t) + C_0 w(t)) + \cdots + (A_d x(t) + B_d y(t) + C_d w(t))t^d \equiv 0. \tag{5}$$

For example, the equations

$$x \cdot w(t) - w \cdot x(t) = 0, \quad y \cdot w(t) - w \cdot y(t) = 0,$$

or equivalently,

$$X \cdot w(t) - x(t) = 0, \quad Y \cdot w(t) - y(t) = 0,$$

are two moving lines of degree $2m$ that follow the rational curve (3). Thus the standard way to find the implicit equation of the rational curve (3) is to compute the resultant of these two moving lines of degree $2m$ that follow the curve. To simplify the determinant that represents the implicit equation, we are going to take the resultant of two moving lines of degree m that follow the curve.

By equating the coefficients of the powers of t in (5) to zero, we obtain $2m + d + 1$ equations in $3d + 3$ unknowns. The $3d + 3$ unknowns $A_0, B_0, C_0, \cdots, A_d, B_d, C_d$ of the moving line (4) can be found by solving the $(2m + d + 1) \times (3d + 3)$ linear system

$$\begin{aligned} \texttt{Coeff}(\, x(t), \quad y(t), \quad w(t), \quad \cdots, \quad x(t)t^d, \quad y(t)t^d, \quad w(t)t^d \,) \\ \cdot \, [\, A_0 \quad B_0 \quad B_0 \quad \cdots \quad A_d \quad B_d \quad C_d \,] = 0, \end{aligned}$$

where "Coeff" stands for the matrix whose columns are the coefficients of the given polynomials. When $d = m$, the dimension of the linear system is $(3m+1) \times (3m+3)$. Consequently, there are at least two linearly independent solutions $p(x, y, w; t)$ and $q(x, y, w; t)$.

The $2m \times 2m$ Sylvester matrix $Syl(p, q)$ obtained by eliminating t from p and q can be written as

$$Syl(p,q) = \texttt{Coeff}\,(\, p, \quad q, \quad pt, \quad qt, \quad \cdots, \quad p\,t^{m-1}, \quad q\,t^{m-1} \,).$$

Theorem 1. *$|Syl(p, q)| = 0$ is the implicit equation of the rational curve (3) when there are no moving lines of degree $< m$ that follow curve (3).*

Proof: Since the implicit equation of a rational curve of degree $2m$ is represented by an irreducible polynomial of degree $2m$ [8], we need only establish three facts:

1) $|Syl(p, q)| \not\equiv 0$,
2) $|Syl(p, q)|$ is of degree at most $2m$,
3) $|Syl(p, q)|$ vanishes on $(x(t) : y(t) : w(t))$.

From the properties of resultants, we know that $|Syl(p, q)| \equiv 0$ if and only if p and q have a common factor $g(t)$ of degree ≥ 1. Since p and q are of degree 1 in x, y, w, one of g and p/g would be of degree 1 in x, y, w, i.e. a moving line of degree $< m$ that follows the curve. But by assumption there are no such moving lines, so $|Syl(p, q)|$ cannot vanish identically.

Since $|Syl(p,q)|$ is the determinant of a $2m \times 2m$ matrix with linear entries in x, y, w, obviously the degree of $|Syl(p,q)|$ is at most $2m$ in x, y, w. Finally, $p(x,y,w;t)$ and $q(x,y,w;t)$ follow the rational curve, so

$$p(x(t_0),y(t_0),w(t_0);t_0) \equiv 0, \qquad q(x(t_0),y(t_0),w(t_0);t_0) \equiv 0,$$

for any parameter t_0. That is, the two polynomials

$$p(x(t_0),y(t_0),w(t_0);t), \qquad q(x(t_0),y(t_0),w(t_0);t)$$

have a common root t_0. Hence, the resultant

$$|Syl(p(x(t_0),y(t_0),w(t_0);t),\ q(x(t_0),y(t_0),w(t_0);t))| = 0.$$

Therefore, $|Syl(p,q)|$ vanishes on $(x(t) : y(t) : w(t))$. □

In summary, we have shown that for a degree $2m$ rational curve, the $2m \times 2m$ Sylvester determinant of two degree m moving lines is the implicit equation of the curve if there are no moving lines of degree $< m$ that follow the curve. The existence of a moving line of degree $m-1$ that follows the curve is equivalent to the vanishing of the $3m \times 3m$ determinant

$$|\texttt{Coeff}(\, x(t), \quad y(t), \quad w(t), \quad \cdots, \quad t^{m-1}x(t), \quad t^{m-1}y(t), \quad t^{m-1}w(t)\,)|\,.$$

This determinant is a polynomial in the coefficients of $x(t), y(t), w(t)$ and therefore almost never vanishes. However, in case such lower degree moving lines do exist, the desired Sylvester determinant can be salvaged by finding the μ–basis (see Section 3).

2.2 An example

Consider the rational sextic curve

$$x(t) = 1 + 2t^2 + 2t^5, \quad y(t) = 2 + t^6, \quad w(t) = 1 + t + 2t^2 + 2t^3 + t^4 + t^6.$$

To use the standard method to implicitize this curve, we introduce two auxiliary polynomials

$$X \cdot w(t) - x(t), \quad Y \cdot w(t) - y(t).$$

Their Sylvester resultant is the 12×12 determinant

$$\begin{vmatrix}
-1+X & -2+Y & 0 & 0 & \cdots & 0 & 0 \\
X & Y & -1+X & -2+Y & \cdots & 0 & 0 \\
-2+2X & 2Y & X & Y & \cdots & 0 & 0 \\
2X & 2Y & -2+2X & 2Y & \cdots & 0 & 0 \\
X & Y & 2X & 2Y & \cdots & 0 & 0 \\
-2 & 0 & X & Y & \cdots & -1+X & -2+Y \\
X & -1+Y & -2 & 0 & \cdots & X & Y \\
0 & 0 & X & -1+Y & \cdots & -2+2X & 2Y \\
0 & 0 & 0 & 0 & \cdots & 2X & 2Y \\
0 & 0 & 0 & 0 & \cdots & X & Y \\
0 & 0 & 0 & 0 & \cdots & -2 & 0 \\
0 & 0 & 0 & 0 & \cdots & X & -1+Y
\end{vmatrix},$$

where the six columns in the middle have been omitted. The Bézout resultant is the 6×6 determinant [4,1]

$$\begin{vmatrix}
2X-Y & -4+4X & 4X-2Y & 2X-Y & -4+2Y & 1+X-Y \\
-4+4X & 4X & 2X-Y & -4+2Y & 1+X+Y & -X \\
4X-2Y & 2X-Y & -4-2Y & 1+X-Y & -X+4Y & 2-2X-2Y \\
2X-Y & -4+2Y & 1+X-Y & -X+4Y & 2-2X+2Y & -2X \\
-4+2Y & 1+X+Y & -X+4Y & 2-2X+2Y & -2X+2Y & -X \\
1+X-Y & -X & 2-2X-2Y & -2X & -X & 2-2Y
\end{vmatrix}.$$

On the other hand, using linear algebra, it is easy to calculate two moving lines of degree three following this curve:

$$\begin{aligned}
&(855w + 31x - 443y) + t\,(77y - 778w - 231x) \\
&\qquad + t^2(338x - 666y) + t^3(25w + 333x - 25y) = 0, \\
&(780w - 413y + 46x) + t\,(-748w + 82y - 196x) \\
&\qquad + t^2(25w - 631y + 333x) + t^3(303x) = 0.
\end{aligned}$$

The new method computes the implicit equation for this curve by taking the 6×6 Sylvester determinant of these two moving lines:

$$\begin{vmatrix}
-443y + 31x + 855w & 46x - 413y + 780w & 0 \\
-778w - 231x + 77y & -748w + 82y - 196x & -443y + 31x + 855w \\
338x - 666y & 25w - 631y + 333x & -778w - 231x + 77y \\
25w + 333x - 25y & 303x & 338x - 666y \\
0 & 0 & 25w + 333x - 25y \\
0 & 0 & 0
\end{vmatrix}$$

$$\begin{vmatrix}
0 & 0 & 0 \\
46x - 413y + 780w & 0 & 0 \\
-748w + 82y - 196x & -443y + 31x + 855w & 46x - 413y + 780w \\
25w - 631y + 333x & -778w - 231x + 77y & -748w + 82y - 196x \\
303x & 338x - 666y & 25w - 631y + 333x \\
0 & 25w + 333x - 25y & 303x
\end{vmatrix}.$$

Using Mathematica, we verified that all three methods produce the correct implicit equation for the given rational curve. Notice that the determinant generated by the new method has the structure of the Sylvester resultant but the order of the Bézout resultant.

2.3 Odd degree rational curves

For a rational curve of degree $2m + 1$, there is always at least one non-zero moving line of degree m and at least 3 linearly independent moving lines of degree $m + 1$ that follow the curve. Therefore, there always exists a moving line p of degree m and a moving line q of degree $m + 1$, where q is not a multiple of p, that follow the rational curve. Suppose there is no moving line of degree $< m$ that follows the curve. Then by an argument similar to the case of even degrees, the Sylvester resultant of p and q is the determinant of a $(2m + 1) \times (2m + 1)$ matrix that represents the implicit equation of the rational curve.

§3. Anti-Annihilation by μ–Basis

The implicitization method in Section 2 works when there are no low degree moving lines that follow the curve. In the rare cases when there do exist low degree moving lines following the curve, the Sylvester resultant used in Section 2 generally vanishes identically [7]. In order to circumvent this difficulty— that is, to counter the annihilation effect of low degree moving lines— and show how the desired Sylvester-style/Bézout-size determinant can still be obtained, we need the notion of a μ–basis [3].

Consider a degree n rational curve $(x(t) : y(t) : w(t))$. By solving an $(n+d+1) \times (3d+3)$ linear system [Section 2.1], we find that the number of linearly independent degree d moving lines that follow this curve is at least $(3d+3) - (n+d+1) = 2d+2-n$. Thus the system always has solutions when $3d+3 > n+d+1$ or $d > n/2-1$. Hence if μ is the lowest degree in t of all the moving lines that follow the curve, then $\mu \leq \lfloor n/2 \rfloor$. Let p be a moving line with the lowest degree μ that follows the curve.

By our previous analysis, there are at least $2(n-\mu)+2-n = n+2-2\mu$ linearly independent moving lines of degree $n-\mu$ that follow the curve. Not all of them can be multiples of p because p can only generate $n+1-2\mu$ independent moving lines of degree at most $n-\mu$: $p, \cdots, pt^{n-2\mu}$. Hence there is a degree $n-\mu$ moving line q that is not a multiple of p.

The two moving lines p and q that we just constructed have the following nice property:

Theorem 2. *Any degree d moving line l that follows the curve $(x(t) : y(t) : w(t))$ can be written uniquely as $Ap + Bq$, where A is a polynomial in t of degree at most $d-\mu$, and B is a polynomial in t of degree at most $d+\mu-n$ [3].*

Proof: [3] presents a proof of this result based on ideal theory. Here we provide a simpler proof using only linear algebra. A degree d moving line can always be written as

$$l = l_x(t)x + l_y(t)y + l_w(t)w,$$

where l_x, l_y, l_w are polynomials in t of degree at most d. It will be very convenient in the following discussion to treat a moving line l as a vector $\vec{l} = (l_x(t), l_y(t), l_w(t))$. Furthermore, note that since the components of the vector $\vec{l}$ are polynomials, the scalar field is the field of rational functions in t.

Let

$$\vec{r} = (x(t), y(t), w(t)), \quad \vec{p} = (p_x(t), p_y(t), p_w(t)), \quad \vec{q} = (q_x(t), q_y(t), q_w(t)).$$

Since, by assumption, the dot products $\vec{p} \cdot \vec{r} = \vec{q} \cdot \vec{r} = 0$, the vector $\vec{r}$ is proportional to the cross product $\vec{p} \times \vec{q}$. That is,

$$\vec{r} = \frac{u(t)}{v(t)}\, \vec{p} \times \vec{q} \tag{6}$$

where $\gcd(u, v) = 1$. Since at least one component of $\vec{r}$ is of degree n and all the components of $\vec{p} \times \vec{q}$ are of degree at most n, the degree of u is at least as great as the degree of v. Moreover, by (6), $u(t)$ divides each component of $\vec{r}$; thus $u(t)$ divides $\gcd(x(t), y(t), w(t)) = 1$. Therefore, the degree of u and the degree of v are both zero, so

$$\vec{r} = \lambda\, \vec{p} \times \vec{q}, \tag{7}$$

where λ is a constant.

Let l be a degree d moving line following the curve. We have $\vec{l} \cdot \vec{r} = 0$ and $\vec{p} \cdot \vec{r} = 0$, $\vec{q} \cdot \vec{r} = 0$, Thus $\vec{l}$, $\vec{p}$, $\vec{q}$ are linearly dependent. Since p and q are linearly independent, we can write

$$\vec{l} = A(t)\vec{p} + B(t)\vec{q}, \tag{8}$$

where $A(t)$ and $B(t)$ are rational functions in t. By (8) and (7), we have

$$\vec{l} \times \vec{q} = A(t)\vec{p} \times \vec{q} = \frac{A(t)}{\lambda}\vec{r}. \tag{9}$$

If $A(t)$ is a polynomial, its degree is at most $d - \mu$ because all the components of $\vec{l} \times \vec{q}$ are of degree at most $d + n - \mu$ and at least one component of $\vec{r}$ is of degree n.

Next we show that $A(t)$ is indeed a polynomial. Since $\gcd(x(t), y(t), w(t))$ is equal to 1, there exist polynomials $x^*(t)$, $y^*(t)$, $w^*(t)$ such that

$$x(t)x^*(t) + y(t)y^*(t) + w(t)w^*(t) = 1.$$

Let $\vec{r}^* = (x^*(t), y^*(t), w^*(t))$. Applying $\vec{r} \cdot \vec{r}^* = 1$ to (9), we have

$$A(t) = \lambda \vec{l} \times \vec{q} \cdot \vec{r}^*. \tag{10}$$

Since the components of all the vectors on the right hand side of (10) are actually polynomials rather than rational functions in t, $A(t)$ must also be a polynomial in t.

The fact that $B(t)$ is a polynomial of degree at most $d + \mu - n$ can be established similarly.

Finally we show that $A(t)$ and $B(t)$ are unique for any given $\vec{l}$. Suppose we have

$$A_1(t)\vec{p} + B_1(t)\vec{q} = A_2(t)\vec{p} + B_2(t)\vec{q};$$

then $(A_1(t) - A_2(t))\vec{p} = (B_2(t) - B_1(t))\vec{q}$. If $B_2(t) - B_1(t)$ is not zero, then it divides $A_1(t) - A_2(t)$, otherwise $p = \vec{p} \cdot (x, y, w)$ is not a moving line with minimal degree that follows the curve. But this would mean that $\vec{q}$ is a multiple of $\vec{p}$; hence, the moving line q is a multiple of the moving line p, which is contrary to assumption. Therefore $A_1(t) = A_2(t)$ and $B_1(t) = B_2(t)$. □

The two moving lines p and q in Theorem 1 are called a $\mu-$basis of the curve $(x(t) : y(t) : w(t))$.

We have shown in Section 2.1 that when there are no moving lines of degree $< m$ following a rational curve of degree $n = 2m$, there will be two moving lines of degree m following the curve and their Sylvester determinant gives the implicit equation. Clearly these two moving lines are simply p and q in Theorem 1 with $\mu = m = n - \mu$. Theorem 1 also tells us that for $\mu \leq d < n - \mu$, a degree d moving line l has the form

$$l = c_0 p + c_1 pt + \cdots c_{d-\mu} pt^{d-\mu},$$

where c_i are constants because l is of the form $Ap + Bq$ with $B = 0$ due to the degree constraints on A and B. Consequently, the Sylvester determinant of any two of these degree d moving lines vanishes as both are multiples of p; furthermore, the number of such linearly independent degree d moving lines is $N_d = d - \mu + 1$. In particular, when there are moving lines of degree $< m$ that follow the curve, we have $\mu < m < n - \mu$, so the Sylvester resultant of any two degree m moving lines vanishes and there are $N_m = m - \mu + 1 \geq 2$ degree m moving lines following the curve. Note that we can find μ in terms of N_m:

$$\begin{aligned} \mu &= m - N_m + 1, \\ N_m &= 3m + 3 - \mathit{Rank\ of} \\ &\qquad \texttt{Coeff}(\, x(t),\, y(t),\, w(t),\, \cdots,\, t^m x(t),\, t^m y(t),\, t^m w(t)\,). \end{aligned}$$

In general then, for a degree n rational curve $(x(t) : y(t) : w(t))$, we can obtain the $\mu-$basis functions p and q by straightforward linear algebra. Since p is irreducible (by degree minimality) and q is not a multiple of p, they have no common factors. Hence their Sylvester resultant

$$Syl(p, q) = \texttt{Coeff}\,(\, p, \quad pt, \quad \cdots, \quad pt^{n-\mu-1}, \quad q, \quad qt, \quad \cdots, \quad qt^{\mu-1}\,)$$

is a matrix of size $n \times n$ whose determinant does not vanish identically. By the divisibility and degree argument of Section 2.1, we see that this Sylvester determinant gives an implicit expression for the rational curve $(x(t) : y(t) : w(t))$ in the style of Sylvester with the order Bézout.

As an example, consider the degree n rational curve

$$(x(t) : y(t) : w(t)) = (1 : t^{n-1} : 1 + t^n). \tag{11}$$

Simple calculations reveal that $p = x + ty - w$ and $q = t^{n-1}x - y$. The Sylvester determinant

$$|\, p \quad \cdots \quad pt^{n-2} \quad q\,| = \begin{vmatrix} x-w & & & & -y \\ y & \ddots & & & \vdots \\ & \ddots & x-w & & \vdots \\ & & y & x & \end{vmatrix} = (-1)^{n-1} y^n + x(x-w)^{n-1}$$

is easily seen to represent the implicit equation of this rational curve.

Acknowledgments. Eng-Wee Chionh is supported by the National University of Singapore for research at Brigham Young University. He greatly appreciates the hospitality and facilities generously provided by BYU. Ming Zhang and Ron Goldman are partially supported by NSF grant CCR-9712345.

References

1. Chionh, E. W., M. Zhang, and R. N. Goldman, Fast computation of the Bézout and Dixon resultant matrices, submitted to Journal of Applied & Computational Mathematics, 1999.
2. Chionh, E. W., M. Zhang, and R. N. Goldman, Surface implicitization using moving planes, in preparation.
3. Cox, D., T. W. Sederberg, and F. Chen, The moving line ideal basis for planar rational curves, Computer Aided Geometric Design **15** (1998), 803–827.
4. Goldman, R. N., T. W. Sederberg, and D. Anderson, Vector elimination: a technique for the implicitization, inversion, and intersection of planar parametric rational polynomial curves, Computer Aided Geometric Design **1** (1984), 327–356.
5. Manocha, D., and J. F. Canny, Implicit representation of rational parametric surfaces, J. Symbolic Computation **13** (1992), 485–510.
6. Salmon, G., *Lessons Introductory to the Modern Higher Algebra*, G. E. Stechert & Co., New York, 1924.
7. Sederberg, T. W., and F. Chen, Implicitization using moving curves and surfaces, *Proceedings of SIGGRAPH*, 1995, 301–308.
8. Sederberg, T. W., R. N. Goldman, and H. Du, Implicitizing rational curves by the method of moving algebraic curves, J. Symbolic Computation **23** (1997), 153–175.
9. Van Der Waerden, B. L., *Modern Algebra Second Edition*, New York, Frederick Ungar, 1950.

Eng-Wee Chionh
School of Computing
National University of Singapore
Lower Kent Ridge Road, Singapore 119260
chionhew@comp.nus.edu.sg

Ming Zhang and Ronald Goldman
Department of Computer Science
Rice University
Houston, TX 77025, USA
mzhang@cs.rice.edu, rng@cs.rice.edu

Better Pasting via Quasi-Interpolation

Blair Conrad and Stephen Mann

Abstract. Surface pasting is a hierarchical modeling technique that places feature surface atop base surface with approximate continuity. In this paper, we describe the use of quasi-interpolation to set the boundary control points of the pasted feature. To achieve interactive performance for surface pasting, we modify the coefficients and weights of quasi-interpolation and devise an efficient sampling scheme. The new surface pasting technique is actually faster than the original surface pasting method when using the same number of feature control points, with a much lower discontinuity between the feature and base surface.

§1. Introduction

Spline curves and surfaces are used in many areas of computer graphics and computer aided geometric design. In particular, tensor product B-spline surfaces are commonly used in modeling and computer animation because they have many attractive properties, such as a compact representation and adjustable levels of internal continuity [7].

Frequently, the user of a piece of modeling or animation software will want to add a region of local detail to a B-spline tensor product surface, but the knot structure will be too coarse to allow the fine-grained control that the user desires. Traditional methods of increasing the complexity of the surface include inserting knots using either Boehm's algorithm [3] or the Oslo algorithm [6]. The insertion of a knot into either of a surface's knot vectors causes an entire row or column of subpatches to be split — rather than increasing the number of subpatches locally, extra subpatches are created across the width or breadth of the surface.

Forsey and Bartels [8] developed hierarchical B-splines to allow the user to add local detail to a tensor product B-spline surface while maintaining B-spline continuity. Hierarchical B-splines suffer from several drawbacks: the regions of added detail must remain parametrically aligned with the base, and it is impractical to slide the features or to maintain a library of overlays that may be added to a base surface. Wavelets [10] can be used in a similar

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 27–36.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

hierarchical modeling technique, but they also require the detail regions to be parametrically aligned with the base.

Displacement mapping is another technique for adding local detail to a surface while adding as few extra control points as possible. Every point on the feature is defined by a displacement vector relative to a certain point in a reference plane. To evaluate the surface at the detail, you add the displacement vector of the feature to the corresponding point on the base surface. The greatest disadvantage to using displacement mapped surfaces is the computational cost. Multiple surface evaluations must be used to determine each point to be rendered on the composite surface.

Surface pasting, suggested by Forsey and Bartels [2], is a generalization of hierarchical B-splines that was intended to combine the flexibility of displacement mapped surfaces with the speed of evaluation enjoyed by hierarchical B-splines. This technique has the flexibility of displacement mapping, but it is much cheaper since only the control points of the feature must be mapped, rather than the larger number of surface points to be rendered. This combination of flexibility and speed has drawn the attention of the modeling industry to surface pasting — recent versions of *Houdini*, a commercial animation tool produced by Side Effects Software, have included support for surface pasting.

However, surface pasting is only an approximation, and as such it does not have the same continuity properties as displacement mapping or hierarchical B-splines. In general, there is no guaranteed continuity between the feature and the base surfaces. By inserting knots into the feature surface, the error between the feature boundary and the base surface can be reduced to any desired tolerance, but many knot insertions may be required to get the desired approximation, and the resulting additional control points in the feature dramatically increase the cost of the pasting operation.

In this paper, we suggest altering the surface pasting technique to improve the approximate continuity between the feature and base surfaces by using quasi-interpolation to set one or more of the outer rings of feature control points.

§2. Surface Pasting

Since the work in this paper is an extension of surface pasting, we will give a quick review of how pasting works. For details on standard surface pasting, see any of the earlier works on the subject [1,2].

The pasting process is illustrated in Figure 1. Surface pasting combines a base surface and a feature surface, each of which is in tensor-product B-spline form. The feature's control points are adjusted so that the boundary of the pasted feature lies on or near the base surface, and the shape of the pasted feature reflects the original shape of the feature imposed as a displacement of the base surface.

To map the feature's control points, the feature's domain is embedded in the feature's range (upper left of Figure 1). Next, a local coordinate frame $\mathcal{F}_{i,j} = \{u_{i,j}, v_{i,j}, w_{i,j}, \mathcal{O}_{i,j}\}$ is constructed for each feature control point $P_{i,j}$

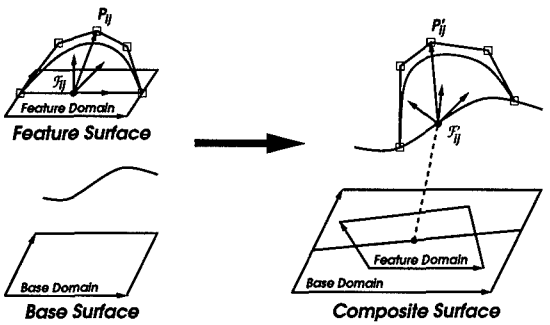


Fig. 1. Surface Pasting.

where the origin $\mathcal{O}_{i,j}$ of each frame is the Greville point corresponding to $P_{i,j}$, two of the frame's basis vectors are the parametric domain directions and the third basis vector is the direction perpendicular to the domain. Each control point $P_{i,j}$ is then expressed relative to its local coordinate frame $\mathcal{F}_{i,j}$ as $P_{i,j} = \alpha u_{i,j} + \beta v_{i,j} + \gamma w_{i,j} + \mathcal{O}_{i,j}$.

Next, we associate the feature's domain with a region in the base's domain (right half of Figure 1). This gives the location on the base surface where we will place the feature. We now map each coordinate frame $\mathcal{F}_{i,j}$ onto the base surface, giving a new coordinate frame $\mathcal{F}'_{i,j} = \{u'_{i,j}, v'_{i,j}, w'_{i,j}, \mathcal{O}'_{i,j}\}$ whose origin $\mathcal{O}'_{i,j}$ is the evaluation of the base surface at $\mathcal{O}_{i,j}$, and two of its basis vectors lie in the tangent plane of the base surface at that point, the third being perpendicular to the tangent plane. We then use the coordinates of each feature control point $P_{i,j}$ relative to $\mathcal{F}_{i,j}$ as coordinates of the elements of the frame $\mathcal{F}'_{i,j}$. This gives us the location of the pasted feature control point, $P'_{i,j} = \alpha u'_{i,j} + \beta v'_{i,j} + \gamma w'_{i,j} + \mathcal{O}'_{i,j}$.

§3. Feature Boundaries

With standard surface pasting, the boundary control points of the pasted feature surface (Figure 2) are conventionally placed on the base surface, resulting in an almost C^0 join. Likewise, the second layer of feature control points are also conventionally located on the base surface, giving an almost C^1 join. By inserting knots into the feature surface, the discontinuity between the feature and the base can be made as small as desired.

One disadvantage of using knot insertion to decrease the discontinuity between the feature surface and the base surface is that it greatly increases the number of control points in the feature surface. In addition to increasing the storage requirements of the feature surface, this increases the computational cost for interactive modeling, as each feature control point needs to be mapped with the pasting process described in the previous section.

The goal of this work is to find better settings of the boundary layers of the feature control points to minimize the C^0 and C^1 discontinuities. Thus, we will find better settings of the control points illustrated in Figure 2. In this figure,

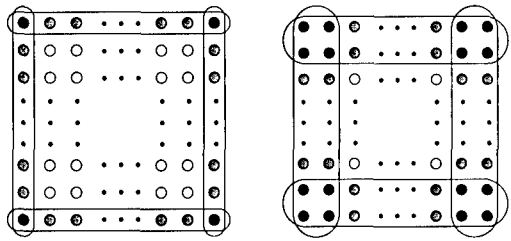


Fig. 2. Approximate C^0 boundary points and C^1 boundary points.

the circled points are set using quasi-interpolation: the black control points will be set to interpolate position and derivative information sampled from the base surface; the gray control points will be set using quasi-interpolation; and the remaining control points will be set using the standard surface pasting process.

§4. Quasi-Interpolation

Quasi-interpolation is a spline approximation technique developed by de Boor and Fix [4]. Their method, given a function f defined over a region of $\mathcal{R}$ and a partition π of $\mathcal{R}$, constructs a degree m spline, $F_\pi f$, that approximates f. $F_\pi f$ is called the quasi-interpolant of f. The quasi-interpolant is a local approximation in that its value at u depends only on the values of f in a small neighborhood around u, it reproduces polynomials of degree m or less, and it provides a high order approximation to f, with $|F_\pi f - f|$ being $\mathcal{O}(|\pi|^{m+1})$.

Lyche and Schumaker [9] describe a family of quasi-interpolation operators Q which include the one of de Boor and Fix. They take

$$Qf = \sum_{i=0}^{M} \lambda_i f B_i^m, \tag{1}$$

where the $\{B_i^m\}_{i=0}^M$ are the B-splines and $\{\lambda_i\}_{i=0}^M$ are linear functionals. The linear functionals can be chosen so that Q is applicable to a wide class of functions, Q is local, and Qf approximates smooth functions with a high order of accuracy. Then the operator Q can be applied to a real-valued function f to produce a B-spline curve Qf that approximates f.

The quasi-interpolant Qf approximates a smooth function f with a high order of accuracy because Q is constructed specifically to reproduce polynomials. The Lyche-Schumaker quasi-interpolants approximate smooth functions with accuracy of up to $\mathcal{O}(h^{m+1})$ for a function with a sufficiently high degree of continuity. In addition, quasi-interpolants can be constructed using linear functionals that can be quickly evaluated, resulting in a fast approximation operator. The combination of high accuracy and speed make quasi-interpolation a fitting tool to improve the approximate continuity around the boundary of pasted features.

The quasi-interpolant Qf is in B-spline form. Each control point of Qf is a weighted sum of linear functionals applied to f:

$$P_i = \sum_{j=0}^{m} \alpha_{i,j} \lambda_{i,j} f.$$

One particular choice of the Lyche-Schumaker quasi-interpolants corresponds to

$$\lambda_{i,j} f = [\tau_{i,0}, \tau_{i,1}, ..., \tau_{i,j}] f,$$

where the τ's are locations at which to evaluate f, and $\alpha_{i,j}$ is the blossom [11] of

$$p_{i,j}(u) = (u - \tau_{i,0})(u - \tau_{i,1}) \cdots (u - \tau_{i,j-1})$$

evaluated at $u_i, ..., u_{i+m-1}$, where the u_i are the knots of Qf. This results in a quasi-interpolant where $Qf = f$ when f is a degree m or lower polynomial and if $Qf \neq f$, then the error has the best possible order.

§5. Modified Quasi-interpolant

The Lyche-Schumaker quasi-interpolant uses cheap coefficients and expensive linear functionals. For interactive modeling using surface pasting, the linear functionals are recalculated frequently (i.e., when we slide the feature across the base) and the coefficients are recalculated less frequently (only when we perform knot insertion to reduce the C^0 discontinuity). Thus, to match the quasi-interpolant to our application, we made new, cheaper linear functionals,

$$\lambda_{i,j} f = f(\tau_{i,j}).$$

This choice of $\lambda_{i,j}$ results in more expensive $\alpha_{i,j}$, which are now the blossom of

$$p_{i,j}(u) = \prod_{k \neq j} \frac{u - \tau_{i,k}}{\tau_{i,j} - \tau_{i,k}} \tag{2}$$

evaluated at $u_i, ..., u_{i+m-1}$, where the u_i are the knots of Qf. Note that the Lyche-Schumaker linear functionals require computing a divided difference of the samples while ours do not, but the denominator term of (2) makes our α coefficients more expensive than theirs.

To use the quasi-interpolant on all four boundaries of a feature, the ends of the connecting boundary curves must match. This is most easily achieved by modifying Q to reproduce position and (to use the quasi-interpolant on the cross-boundary derivatives) d derivatives at its endpoints.

Thus, we devised a quasi-interpolant that we call Q^d, where the linear functionals for the control points near the ends of the curves are derivatives of the original curve at the endpoints. This quasi-interpolant is actually a special case of the Lyche-Schumaker quasi-interpolant. The Q^d quasi-interpolant requires different λ's and p's at the endpoints, which in turn require new α's.

For $0 \le i \le M$, and $0 \le j \le m$, we define Q^d as

$$Q^d f = \sum_{i=0}^{M} P_i B_i^m = \sum_{i=0}^{M} \Big(\sum_{j=0}^{m} \alpha_{i,j} \lambda_{i,j} f \Big) B_i^m, \tag{3}$$

where

$$\lambda_{i,j} f = \begin{cases} f(\tau_{i,j}), & \text{if } d < i < M-d \\ D^j f(u_{m-1}), & \text{if } i \le d \text{ and } j \le i \\ D^j f(u_M), & \text{if } i \ge M-d \text{ and } j \le M-i \\ 0, & \text{otherwise} \end{cases}$$

$$p_{i,j}(u) = \begin{cases} \displaystyle\prod_{k \ne j} \frac{u - \tau_{i,k}}{\tau_{i,j} - \tau_{i,k}}, & \text{if } d < i < M-d \\ \dfrac{(u-u_{m-1})^j}{j!}, & \text{if } i \le d \text{ and } j \le i \\ \dfrac{(u-u_M)^j}{j!}, & \text{if } i \ge M-d \text{ and } j \le M-i \end{cases}$$

$$\alpha_{i,j} = \begin{cases} 0, & \text{if } i \le d \text{ and } j > i \\ 0, & \text{if } i \ge M-d \text{ and } j > M-i \\ p_{i,j}(u_i, \ldots, u_{i+m-1}), & \text{otherwise} \end{cases}$$

where the $\tau_{i,j}$s are as defined in the next section. Note that for $d < i < M-d$, the $p_{i,j}$s are merely Lagrange polynomials. This quasi-interpolant Q^d interpolates the position and derivatives at the endpoints and has optimal error bounds as discussed by Conrad [5].

Sampling discipline

The above equations require us to sample the function f to be approximated at some values $\tau_{i,j}$, and (3) suggests that we require about $m+1$ samples of the base surface for each boundary control point of the feature. Since the quasi-interpolant error bounds hold for most choices of τ's, we managed to reduce the number of samples of the base surface by selecting the $\tau_{i,j}$ in the following way:

- Place the first d and last d control points in groups of their own; divide the remaining control points into groups as indicated by the pseudo-code of Figure 3; each group k will contain control points $P_{N_k}, \ldots, P_{N_k+n_k-1}$, where $N_k = \sum_{\ell=0}^{k-1} n_\ell$ and n_k is the number of control points in group k;
- Let γ_i be the Greville point associated with control point P_i; then for each group k, choose the interval from which to sample as
$$[a_k, b_k] = [(\gamma_{N_k-1} + \gamma_{N_k})/2, (\gamma_{N_{k+1}-1} + \gamma_{N_{k+1}})/2];$$
- Set $\tau_{i,j} = a_k + j(b_k - a_k)/m$, for $i = N_k, \ldots, N_{k+1} - 1$ and $j = 0, \ldots, m$;
- Sample f at the $\tau_{i,j}$; note that you can share the first and last samples between groups,

$I := M - 2 * d + 1$
$numFullGroups := \lfloor I/m \rfloor$
$L := I - m * numFullGroups$
if $L = 0$
    make $numFullGroups$ groups of m
else if $numFullGroups \equiv 0 \pmod 2$
    put leftovers in middle group; all others have m
else there are an odd number of full groups
    if $L \equiv 0 \pmod 2$
        first and last groups contain $L/2$ each; all others
    else
        move one control vertex from middle full group to the leftovers
        the first and last groups each contain $(L + 1)/2$,
        the middle group contains $m - 1$, and the others contain m

Fig. 3. Pseudo-code for grouping.

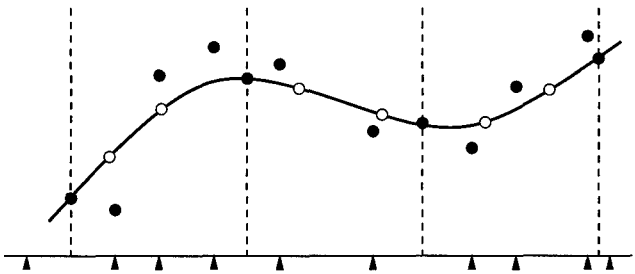


Fig. 4. Sampling discipline for quasi-interpolated surface pasting.

Pseudo-code for our grouping scheme appears in Figure 3, and the sampling is illustrated in Figure 4. The white and gray points are samples of the base surface; the black points represent the control points for the quasi-interpolant (the actual quasi-interpolant control points will lie much closer to the curve; we have moved them away from their actual position for illustrative purposes); the dotted lines indicate the boundaries between the groups, with the gray points being shared by two groups; and the triangles indicate the position of the Greville points.

This sampling method reduces the number of base surface evaluations to about one sample per control point.

Quasi-interpolated surface pasting

Ideally, we would be able to integrate quasi-interpolation into surface pasting by running four independent quasi-interpolations to set the boundary control points, then four more independent quasi-interpolations to set the second layer of control points, and so on. However, as seen in Figure 2, the boundary layers

of control points overlap, and while the gray control points in this figure can be set independently, the black ones are shared by two boundaries. To build a tensor-product patch, we need the two quasi-interpolation solutions that compute a group of black points to give these points the same settings.

Our solution is to sample the base surface at the corners of the embedded feature domain for position and as many derivatives as we are trying to approximate along the boundary. In our implementation, we were only trying to achieve approximate C^1 continuity, so we sampled for position and first derivatives (including mixed partial derivatives).

Next, we used the Q^d operator to set $d+1$ outermost rings of control points. With $d = 1$ (the right side of Figure 2), this means that we make eight applications of the Q^1 operator, four for the outer layer of control points, and four for the next layer of control points. In each application, the Q^1 operator interpolates four values (those indicated by the black points) and quasi-interpolates the remaining values.

Note that while we quasi-interpolate points to compute the first layer of control points, to compute the second layer of control points, we first quasi-interpolate cross-boundary derivative vectors. The vector coefficients of this second quasi-interpolant are then scaled and added to the first layer control points to produce the second layer of control points.

We set the remaining interior control points, which do not affect the cross-boundary derivatives, using the standard pasting method.

Results

We integrated the Q^0 and Q^1 quasi-interpolants into surface pasting. In our particular implementation, this results in a cubic quasi-interpolant being used to approximate a C^2 curve. Theoretically, the quasi-interpolant described in this paper should have $\mathcal{O}(h^3)$ order of accuracy, which matched our empirical tests. This is an improvement over standard pasting, which empirically had $\mathcal{O}(h^2)$ accuracy on the same data.

We achieved a similar improvement for the cross-boundary derivatives (improving from $\mathcal{O}(h^1)$ to $\mathcal{O}(h^2)$), and the Q^1 method has an additional advantage over standard pasting: for standard pasting, you insert knots in one parametric direction to decrease the C^0 discontinuity, and insert knots in the other parametric direction to decrease the C^1 discontinuity. With the Q^1 method, inserting knots in one parametric direction will decrease both the C^0 and C^1 discontinuity along the boundary.

Visually, the quasi-interpolation surface pasting method gives significant improvements in the approximate C^d continuity around the feature boundaries. An example appears in Figure 5. In all three images, the feature surface has the same number of control points. The large gap that appears in the standard surface pasting example has almost disappeared with the Q^0 and Q^1 pasted surfaces, but a "corner" has appeared in the Q^0 surface. However, Q^1 pasted surfaces have better approximate continuity (both C^0 and C^1) and are cheaper to paste than standard pasting, assuming we use feature surfaces having the same number of control points for both standard and Q^1 pasting.



Fig. 5. Standard pasted feature, Q^0 pasting, and Q^1 pasting.

This computational gain for Q^1 pasting is a result of quasi-interpolating the cross-boundary vector field when computing the second layer of control points. We obtain the vectors to quasi-interpolate when we evaluate the base surface for quasi-interpolating the boundary curve. Thus, we avoid having to evaluate the base surface when computing the second layer of control points.

The biggest computational gain, however, comes from needing fewer control points for Q^1 feature surfaces than for standard pasting surfaces to achieve the same tolerance along the boundaries. In particular, in the examples we tested, the Q^1 method required one third the number of boundary points as the standard surface pasting method to achieve the same error along the boundary. This results in one-ninth the total number of control points in the feature, resulting in roughly a factor of ten speedup for quasi-interpolated surface pasting over standard surface pasting.

Conclusion

Surface pasting is a flexible modeling technique that allows the creation of multi-resolution surfaces by hierarchically composing tensor product B-spline surfaces. The resulting composite surface is only approximately C^0 and approximately C^1. In this paper, we developed quasi-interpolation operators to reduce the C^0 and C^1 discontinuities between the feature and base surfaces. By reducing the C^0 and C^1 discontinuities, we can use fewer control points in the representation of the feature surfaces, resulting in a significant speedup of the pasting technique.

We implemented the technique described in this paper for pasted surfaces of arbitrary degree, and tested the method on cubic and quartic surfaces. While we have only reduced C^0 and C^1 errors, the extension to higher levels of approximate continuity is straightforward, but calculating the coefficients that weight the linear functionals will be more complicated.

Acknowledgments. Many thanks to Richard Bartels and Kirk Haller, who have provided many valuable insights during the course of our research. We also thank Tom Lyche, who originally suggested the topic of this research. Financial support for this research was provided by NSERC, CITO, and the University of Waterloo.

References

1. Barghiel, C., R. Bartels, and D. Forsey, Pasting spline surfaces, in *Mathematical Methods for Curves and Surfaces*, Morten Dæhlen, Tom Lyche, Larry L. Schumaker (eds), Vanderbilt University Press, Nashville & London, 1995, 31-40.
2. Bartels, R. and D. Forsey, Spline overlay surfaces, University of Waterloo Report CS-92-08, 1991.
3. Boehm, Wolfgang, Inserting new knots into B-spline curves, Computer-Aided Design **12(4)** (1980), 199–201.
4. Boor, C. de and G. J. Fix, Spline approximation by quasi-interpolants, J. Approx. Theory **8** (1973), 19–45.
5. Conrad, B., Better pasting through quasi-interpolation, University of Waterloo Report CS-99-14, 1999.
6. Cohen, E., T. Lyche, and R. Riesenfeld, Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics, Comp. Graphics and Image Proc. **14** (1980), 87–111.
7. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Fourth Edition, Academic Press, NY, 1994.
8. Forsey, D. and R. Bartels, Hierarchical B-spline refinement, Computer Graphics **22(4)** (1988), 205–212.
9. Lyche, T. and L. L. Schumaker, Local spline approximation methods, J. Approx. Theory **15** (1975), 294–325.
10. Stollnitz, E., T. DeRose, and D. Salesin, *Wavelets for Computers Graphics*, Academic Press, Boston, 1996.
11. Ramshaw, L., Blossoming: a connect-the-dots approach to splines, Techn. Rep., Digital Systems Research Center, Palo Alto, 1987.

Blair Conrad and Stephen Mann
Computer Science Department
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
beconrad@cgl.uwaterloo.ca
smann@cgl.uwaterloo.ca

Conjugate Silhouette Nets

W. L. F. Degen

Abstract. Conjugate nets, Laplace transformations and projective translation surfaces are exploited for CAGD purposes. The latter are shown to be equivalent with conjugate nets having degenerated Laplace transforms. Relations to conjugate nets with planar silhouettes, supercyclides and Dupin cyclides are given.

§1. Conjugate Nets and their Laplace Transforms

Conjugate nets play an important role in classical differential geometry, especially because of their projective invariance (see [3]). Representing a surface in d-space by *homogeneous* coordinates

$$\mathcal{S} \quad \ldots \quad \boldsymbol{x} : D \to \mathbb{R}^{d+1},\ D \subset \mathbb{R}^2,\ \ \boldsymbol{x} \in C^\infty[D], \tag{1}$$

where D is an open connected domain of the "parameter plane" $\mathbb{R}^2$, then a conjugate net is defined by the validity of a Laplacian equation

$$\boldsymbol{x}_{uv} + a\boldsymbol{x}_u + b\boldsymbol{x}_v + c\boldsymbol{x} = 0 \tag{2}$$

with certain functions $a, b, c \in C^\infty[D]$. To exclude planar surfaces

$$\dim \operatorname{span}(\boldsymbol{x}, \boldsymbol{x}_u, \boldsymbol{x}_v, \boldsymbol{x}_{uu}, \boldsymbol{x}_{vv}) \geq 3, \quad d \geq 3 \tag{3}$$

is assumed throughout. The geometric meaning of (2) for $d = 3$ is that the characteristic lines of the tangent planes along one isoparametric line are tangent to the other. A second characterization is that the two tangents to the isoparametric lines have a *harmonic cross ratio* with the two asymptotic tangents.

The most important property of conjugate nets is that they have in both isoparametric directions a *Laplace transformed net* (*Laplace transform*) which is defined by the striction points of the two developables generated by the

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 37–44.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

tangent planes along the net curves (isoparametric lines). Analytically, they are given by

$$\mathcal{L}_u(\boldsymbol{x}) \quad \ldots \quad \boldsymbol{y} := \boldsymbol{x}_u + b\boldsymbol{x}, \tag{4}$$

$$\mathcal{L}_v(\boldsymbol{x}) \quad \ldots \quad \boldsymbol{z} := \boldsymbol{x}_v + a\boldsymbol{x}. \tag{5}$$

Indeed, by (2), we get

$$\boldsymbol{y}_v = h\boldsymbol{x} - a\boldsymbol{y}, \tag{6}$$

$$\boldsymbol{z}_u = k\boldsymbol{x} - b\boldsymbol{z} \tag{7}$$

with

$$h = b_v + ab - c, \quad k = a_u + ab - c \tag{8}$$

being the so-called Darboux invariants. Eqn. (6) shows that the line $\boldsymbol{x} \wedge \boldsymbol{y}$ is tangent to the u-curve (isoparametric line with $v = \text{const}$, u varying) on the surface $(\boldsymbol{x})_D$ as well as tangent to the v-curve on $(\boldsymbol{y})_D$; analogously $\boldsymbol{x} \wedge \boldsymbol{z}$ is tangent to the v-curve on $(\boldsymbol{x})_D$ and the u-curve on $(\boldsymbol{z})_D$.

In euclidean differential geometry, conjugate nets owe their importance to the fact that the *curvature lines* on every surface are conjugate (except at umbilic points). Thus, for CAGD applications, one can profit from the theory of conjugate nets since, for instance, the two families of circles on a Dupin cyclide or the net of profile curves and meridian circles on a surface of revolution is a conjugate net.

In this paper we deal with the special case that the two Laplace transforms *degenerate* into curves. Assuming that the net curves have nowhere asymptotic directions, i.e.

$$\boldsymbol{x} \wedge \boldsymbol{x}_u \wedge \boldsymbol{x}_{uu} \neq 0\,, \quad \boldsymbol{x} \wedge \boldsymbol{x}_v \wedge \boldsymbol{x}_{vv} \neq 0, \tag{9}$$

then the degeneration conditions $\boldsymbol{y} \wedge \boldsymbol{y}_u \wedge \boldsymbol{y}_v = 0$ and $\boldsymbol{z} \wedge \boldsymbol{z}_u \wedge \boldsymbol{z}_v = 0$ imply

$$h = 0,\ k = 0 \quad \text{for all } (u, v) \in D. \tag{10}$$

Definition 1. *A conjugate net in 3-space satisfying (10) with regularity conditions (3), (9) is called a* conjugate silhouette net.

This notation is justified since by (4), (6), (10) all u-tangents along a v-curve meet at the *fixed* point $\boldsymbol{y}$, thus building up a general cone with apex $\boldsymbol{y}$. Thus the v-curve $\boldsymbol{x}(u_0, v)$ can be considered as a silhouette on the surface $\mathcal{S}$ by central illumination from $\boldsymbol{y}$. Similarily the u-curves are silhouette lines by central illumination from $\boldsymbol{z}$.

On the other hand, it is easy to see that a net of silhouette lines on a surface whereby the centers of illumination vary on two curves is automatically conjugate, provided that the generators of the enveloping cones are tangent to

the net curves. Thus, the equations (2), (10) characterize conjugate silhouette nets (up to degenerated cases).

At first glance, this class of surfaces seems to be very restricted. But this is not true: it comprises many subclasses of surfaces considered in CAGD literature such as Dupin cyclides ([2,4,7]), supercyclides ([1,5,8,9]), nets with planar silhouettes ([6]) etc.. As for all of these examples, the whole class of conjugate silhouette nets is well-suited for CAGD applications, in particular for geometric modelling purposes because of their simple blending properties: Putting two of them together along a common net curve immediately yields a G^1-continuity, once the corresponding centers of illumination coincide.

But there is still another reason making these surfaces worth considering in CAGD: They admit a very simple generation as so-called "projective translation surfaces", as will be derived in the next section.

§2. Projective Translation Surfaces

Let

$$\mathcal{C}_1 \ \ldots \ \boldsymbol{p} : I_1 \to \mathbb{R}^{d+1}, \ \ \mathcal{C}_2 \ \ldots \ \boldsymbol{q} : I_2 \to \mathbb{R}^{d+1} \tag{11}$$

be two C^∞-curves in d-space represented also in homogeneous coordinates (I_1, I_2 being two open nonvoid intervals of $\mathbb{R}$). Then one gets a surface $\mathcal{S}$ (1) simply by setting

$$\mathcal{S} \ \ldots \ \boldsymbol{x}(u,v) := \boldsymbol{p}(u) + \boldsymbol{q}(v), \quad (u,v) \in I_1 \times I_2 := D \tag{12}$$

Definition 2. *Surfaces defined by (12) via two curves (11) are called* projective translation surfaces.

This definition generalizes the usual euclidean (or affine) definition of translation surfaces, where the same formula (12) is used but interpreted in affine (non-homogeneous) coordinates. So one curve can be considered to move along the other thus sweeping out the surface. In the projective case, the generating point $\boldsymbol{x}(u,v)$ always lies on the line $\boldsymbol{p}(u) \wedge \boldsymbol{q}(v)$ joining these two points of $\mathcal{C}_1$ and $\mathcal{C}_2$ independently. It must be noticed that the *normalizations* are essential (not arbitrarily to be choosen like usually when dealing with curves): they determine the *position* of that point $\boldsymbol{x}(u,v)$ on the line $\boldsymbol{p}(u) \wedge \boldsymbol{q}(v)$.

Now we can establish one of our main results:

Theorem 1. *Every conjugate silhouette net is a projective translation surface, and the net curves correspond to the isoparameter lines in the representation (12).*

Proof: We have, by definition, $h = 0$, $k = 0$, and hence in particular

$$a_u = b_v. \tag{13}$$

Assuming D to be simply connected, we conclude that there exists a C^∞-function $f : D \to \mathbb{R}$ with

$$f_u = b, \quad f_v = a. \tag{14}$$

Taking $\rho := e^f$, we calculate $\rho_u = \rho b$, $\rho_v = \rho a$ and $\rho_{uv} = (\rho b)_v = \rho(ab + b_v) = \rho c$, the latter observing (8), (10). Since $\rho \neq 0$ in D, we can *renormalize* $\overline{\boldsymbol{x}} := \rho \boldsymbol{x}$, obtaining

$$\overline{\boldsymbol{x}}_{uv} = (\rho_{uv} - \rho c)\boldsymbol{x} + (\rho_u - \rho b)\boldsymbol{x}_v + (\rho_v - \rho a)\boldsymbol{x}_u, \tag{15}$$

and thus

$$\overline{\boldsymbol{x}}_{uv} = 0. \tag{16}$$

This equation immediately yields a representation (12) by integration (possibly restricted to a rectangle $I_1 \times I_2$ within D). □

Up to now we think of that renormalization as always having been done, so the Laplace equation (2) has the coefficients

$$a = 0,\ b = 0,\ c = 0. \tag{17}$$

Therefore, the Laplace transforms (4) and (5) are now given by

$$\mathcal{L}_u(\boldsymbol{x}) \ \ldots \ \boldsymbol{y}(u) = \frac{d\boldsymbol{p}(u)}{du}, \qquad u \in I_1, \tag{18}$$

$$\mathcal{L}_u(\boldsymbol{x}) \ \ldots \ \boldsymbol{z}(v) = \frac{d\boldsymbol{q}(v)}{dv}, \qquad v \in I_2. \tag{19}$$

Calling these curves the projective hodographs of $\boldsymbol{p}$ and $\boldsymbol{q}$ respectively, we can state

Corollary. *The Laplace transforms of a conjugate silhouette net $\mathcal{L}$ are the projective hodographs of the generating curves C_1, C_2 of $\mathcal{L}$ (considered as a projective translation surface).*

§3. Axial Silhouette Nets

Definition 3. *A conjugate silhouette net is called* axial *if the generating curves C_1, C_2 in its representation (12) as a projective translation surface are (parts of) straight lines. These lines are called the* first and the second axis *of the net.*

The conditions for axial conjugate silhouette nets are that $\boldsymbol{p}, \boldsymbol{p}', \boldsymbol{p}''$ and likewise $\boldsymbol{q}$, $\boldsymbol{q}'$, $\boldsymbol{q}''$ must be linearly dependent (a prime at $\boldsymbol{p}$ indicating derivation with respect to u and at $\boldsymbol{q}$ with respect to v). Assuming $\boldsymbol{p}$, $\boldsymbol{p}'$ and likewise $\boldsymbol{q}$, $\boldsymbol{q}'$ to be linearly independent (otherwise the point would be stationary) we have

$$\boldsymbol{p}'' = \alpha\boldsymbol{p} + \beta\boldsymbol{p}', \quad \boldsymbol{q}'' = \gamma\boldsymbol{q} + \delta\boldsymbol{q}' \tag{20}$$

with some C^∞-functions α, β of u and γ, δ of v characterizing axial nets.

This has many consequences; most of them we proved earlier for supercyclides and for nets with planar silhouettes [5,6]. Now we give the result for the general case of axial silhouette nets:

Theorem 2.

a) *All the net curves (of both families) are planar curves,*

b) *The planes of the net curves of each family belong to a pencil,*

c) *The axes of these two pencils coincide with the second and the first axis of the net (i. e. the plane of a curve of the first [second] family passes through the second [first] axis,*

d) *The apexes of the envelopping cones (the "light centers") along a u-curve [v-curve] lie on the second [first] axis,*

e) *Any two u-curves [v-curves] are projectively equivalent to each other.*

Proof: We perform the proofs only for the u-curves; the assertions with respect to the v-curves follow analogously.

a): From (12), (20) we derive

$$\boldsymbol{x}_u = \boldsymbol{p}', \quad \boldsymbol{x}_{uu} = \alpha\boldsymbol{p} + \beta\boldsymbol{p}', \tag{21}$$

$$\boldsymbol{x}_{uuu} = (\alpha' + \alpha\beta)\boldsymbol{p} + (\alpha + \beta' + \beta^2)\boldsymbol{p}'. \tag{22}$$

Hence $\boldsymbol{x}_u \wedge \boldsymbol{x}_{uu} \wedge \boldsymbol{x}_{uuu} = 0$, meaning that the u-curves are planar.

b), c): Eqns. (21) show that $\boldsymbol{p}$, $\boldsymbol{p}'$ are contained in the plane $\boldsymbol{x} \wedge \boldsymbol{x}_u \wedge \boldsymbol{x}_{uu}$ of this u-curve; but $\boldsymbol{p}$, $\boldsymbol{p}'$ span the first axis.

d): The apex of the envelopping cone is given by (21) as $\boldsymbol{p}'$. Hence it is lying on the first axis.

e): Assumption (9) implies $\alpha \neq 0$ for all $u \in I_1$. Thus, $\boldsymbol{x}_u$ and $\boldsymbol{x}_{uu}$ can be eliminated from (21) and with this Eqn. (22) yields

$$\boldsymbol{x}_{uuu} = \left(\frac{\alpha'}{\alpha} + \beta\right)\boldsymbol{x}_{uu} + \left(\frac{\alpha'}{\alpha}\beta + \alpha + \beta'\right)\boldsymbol{x}_u. \tag{23}$$

Thus the coefficient of the fundamental equation (see [3]) do not depend on the second parameter v; this means geometrically that all u-curves are projectively equivalent. □

For CAGD purposes *rational* (and polynomial) surfaces are of particular interest. The explicit representation (12) makes it very easy to pick out rational surfaces from that general class: The only thing one has to do is to insert *rational representations* for $\boldsymbol{p}(u)$ and $\boldsymbol{q}(v)$. We restrict this procedure to axial nets, and derive from it the (rational) Bézier representation.

Theorem 3. *For any pair of planar rational curves C_1 and C_2, there exists an axial conjugate silhouette net having its u-curves projectively equivalent to C_1 and its v-curves projectively equivalent to C_2. The axes can be arbitrarily prescribed as two skew straight lines.*

Proof: Let C_1, C_2 be represented in planar homogeneous coordinates by triples of linearly independent polynomials

$$C_1 \ \ldots \ x_i = f_i(u) \ (i = 0, 1, 2), \quad C_2 \ \ldots \ x_i = g_i(v) \ (i = 0, 1, 2). \tag{24}$$

Furthermore, let the axes be spanned by vectors $\boldsymbol{a}$, $\boldsymbol{b} \in \mathbb{R}^4$ and $\boldsymbol{c}$, $\boldsymbol{d} \in \mathbb{R}^4$ resp. Then $det(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{d}) \neq 0$ since the axes are assumed to be skew. With this we can set

$$\boldsymbol{p}(u) := \frac{1}{f_0(u)}\big(f_1(u)\boldsymbol{a} + f_2(u)\boldsymbol{b}\big), \quad \boldsymbol{q}(v) := \frac{1}{g_0(v)}\big(g_1(v)\boldsymbol{c} + g_2(v)\boldsymbol{d}\big) \quad (25)$$

getting the desired axial net $\mathcal{S}$ by (12) (restricted to intervals $I_1, I_2 \subset \mathbb{R}$ where $f_0(u)$ resp. $g_0(v)$ have no zeros). Now, indeed, the u-curves are planar and projectively equivalent to $\mathcal{C}_1$ since, for fixed $v = v_0$, we have $\overline{\boldsymbol{p}}(u) = f_0(u)\,\boldsymbol{p}(u) = f_0(u)\boldsymbol{q}(v_0) + f_1(u)\boldsymbol{a} + f_2(u)\boldsymbol{b}$ so that $f_i(u)$ are the coordinates with respect to basis $\boldsymbol{q}(v_0)$, $\boldsymbol{a}$, $\boldsymbol{b}$. The proof for the v-curves follows the same line, mutatis mutandis. □

Obviously, the representation (25) is not unique. But we can immediately derive from (12) and (25) the Bézier representation of $\mathcal{S}$: First renormalize (12) with the factor $f_0(u)g_0(v)$ getting

$$\overline{\boldsymbol{x}}(u,v) \;=\; g_0(v)\big(f_1(u)\boldsymbol{a} + f_2(u)\boldsymbol{b}\big) \;+\; f_0(u)\big(g_1(v)\boldsymbol{c} + g_2(v)\boldsymbol{d}\big), \quad (26)$$

and then we expand the polynomials $f_i(u)$ and $g_i(v)$ with repsect to the Bernstein basis

$$f_i(u) = \sum_{j=0}^{n} \alpha_{i,j} B_j^n(u), \quad g_k(v) = \sum_{l=0}^{m} \beta_{k,l} B_l^m(v), \quad (27)$$

getting the usual homogeneous Bézier representation

$$\overline{\boldsymbol{x}}(u,v) \;=\; \sum_{j=0}^{n}\sum_{l=0}^{m} \boldsymbol{b}_{j,l} B_j^n(u)\, B_k^m(v) \quad (28)$$

with the control points

$$\boldsymbol{b}_{j,l} = \beta_{0,l}(\alpha_{1,j}\boldsymbol{a} + \alpha_{2,j}\boldsymbol{b}) \;+\; \alpha_{0,j}(\beta_{1,l}\boldsymbol{c} + \beta_{2,l}\boldsymbol{d}). \quad (29)$$

Since these calculations can be done also backwards, we obtain

Corollary. *The conditions (29) for the control points characterize a (n, m)-rational Bézier surface (28) to be an axial conjugate silhouette net.*

§4. Applications to Dupin Cyclides

Dupin cyclides are special kinds of supercyclides [5], and therefore they should have representations as axial conjugate silhouette nets so far they have fourth order and skew axes. However, their usual representation starts from the midpoint curves $\boldsymbol{Y}(u)$ and $\boldsymbol{Z}(v)$ of the two families of envelopping spheres and results in

$$\mathcal{D} \;\ldots\; \boldsymbol{X}(u,v) = \frac{r_2(v)\boldsymbol{Y}(u) + r_1(u)\boldsymbol{Z}(v)}{r_2(v) + r_1(u)}, \quad (30)$$

where r_1, r_2 denote the corresponding radius functions. Though this is also a sum of two vectors it has neither the form (12) nor are the components parts of straight lines.

Thus, the question arises of how to transform (30) into (12). The first step to solve this problem consists in passing to homogeneous coordinates $\boldsymbol{x} = (x_0, x_1, x_2, x_3)^T$

$$\boldsymbol{x} = \rho \begin{pmatrix} 1 \\ \boldsymbol{X} \end{pmatrix}, \quad \rho \in \mathbb{R} \setminus \{0\} \tag{31}$$

(i.e. $x_0 = \rho \cdot 1$, $x_i = \rho \cdot X_i$ $(i = 1, 2, 3)$) and to take in the present case $\rho = \frac{1}{r_1} + \frac{1}{r_2}$ yielding indeed (12)

$$\boldsymbol{p}(u) = \begin{pmatrix} \dfrac{1}{r_1(u)} \\ \dfrac{1}{r_1(u)} \boldsymbol{Y}(u) \end{pmatrix}, \quad \boldsymbol{q}(v) = \begin{pmatrix} \dfrac{1}{r_2(v)} \\ \dfrac{1}{r_2(u)} \boldsymbol{Z}(u) \end{pmatrix}. \tag{32}$$

However, the curves $\boldsymbol{p}$ and $\boldsymbol{q}$ describe an ellipse and a hyperbola as before. Obviously, another representation of the same kind (12) must satisfy

$$\overline{\boldsymbol{p}}(u) = \boldsymbol{p}(u) + \boldsymbol{c}, \quad \overline{\boldsymbol{q}}(v) = \boldsymbol{q}(v) - \boldsymbol{c} \tag{33}$$

with a *constant* vector $\boldsymbol{c}$.

Starting with the explicit representations

$$\boldsymbol{Y}(u) = \begin{pmatrix} 1 \\ \dfrac{1-u^2}{1+u^2}\rho \\ \dfrac{2u}{1+u^2}\rho\omega \\ 0 \end{pmatrix}, \qquad \boldsymbol{Z}(v) = \begin{pmatrix} 1 \\ \dfrac{1+v^2}{1-v^2}\rho\sigma \\ 0 \\ \dfrac{2v}{1-v^2}\rho\omega \end{pmatrix} \tag{34}$$

(with some shape parameters $d, \rho, \sigma, |\sigma| < 1$, $\omega = \sqrt{1-\sigma^2}$) and observing the corresponding radius functions

$$r_1(u) = \frac{f_0(u)}{1+u^2}, \quad r_2 = \frac{g_0(v)}{1-v^2}, \tag{35}$$

whereby

$$f_0(u) = (1+u^2)d\rho - (1-u^2)\rho\sigma, \; g_0(v) = (1+v^2)\rho - (1-v^2)d\sigma,$$

we finally obtain

$$\overline{\boldsymbol{p}}(u) = \frac{1}{f_0(u)} \left(\left((1+u^2) - \frac{1}{d\sigma} f_0(u) \right) \begin{pmatrix} 1 \\ d \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 2\rho\omega u \\ 0 \end{pmatrix} \right) \tag{36}$$

and

$$\overline{\boldsymbol{q}}(u) = \frac{1}{g_0(u)} \left(\left((1-v^2) + \frac{1}{d\sigma} g_0(v)\right) \begin{pmatrix} 1 \\ d\sigma^2 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 2\rho\omega v \end{pmatrix} \right) \tag{37}$$

with

$$\boldsymbol{c} = \frac{1}{d\sigma} (1,\ 0,\ 0,\ 0)^T .$$

Thus we proved

Theorem 4. *The formulas (36), (37) (inserted into (12)) yield an explicit representation of nonparabolic Dupin cyclides with skew axes as axial conjugate silhouette nets with respect to a suitable (homogeneous) coordinate system.*

References

1. Allen, S. and D. Dutta, Supercyclides and blending, Computer Aided Geometric Design **14** (1997), 637–651.
2. Boehm, W., On cyclides in geometric modeling, Computer Aided Geometric Design **7** (1990), 243–255.
3. Bol, G., *Projektive Differentialgeometrie I – III*, Vandenhœck & Ruprecht, Göttingen, 1950 – 1967.
4. Chandru, V., D. Dutta, and C. M. Hoffmann, On the geometry of Dupin cyclides, The Visual Computer **5** (1989), 277–290.
5. Degen, W. L. F., Generalized cyclides for use in CAGD, in *Computer-Aided Surface Geometry and Design, The Mathematics of Surfaces IV*, Adrian Bowyer (ed.) Clarendon Press, Oxford, 1994, 349–363.
6. Degen, W. L. F., Nets with plane silhouettes, in *Design and Application of Curves and Surfaces, The Mathematics of Surfaces V*, R.B. Fisher (ed.) Clarendon Press, Oxford, 1994, 117–133.
7. Pratt, M. J., Cyclides in computer aided geometric design, Computer Aided Geometric Design **7** (1990), 221–242.
8. Pratt, M. J., Dupin cyclides and supercyclides, in *The Mathematics of Surfaces VI*, G. Mullineux (ed.), Oxford, University Press, 1996, 43 – 66.
9. Pratt, M. J., Quartic supercyclides I: Basic theory. Computer Aided Geometric Design **14** (1997), 671–692.

Prof. Dr. W.L.F Degen
University of Stuttgart
Mathematisches Institut B
Pfaffenwaldring 57
D-70569 Stuttgart, Germany
degen@mathematik.uni-stuttgart.de

From PS–splines to NURPS

Paul Dierckx and Joris Windmolders

Abstract. A normalized B–spline representation for Powell–Sabin (PS) spline surfaces is extended to piecewise rational surfaces (NURPS). We investigate the adaptation of existing algorithms operating on B–splines to this more general case, the influence of weights and their geometrical interpretation, the possibility of representing planar sections, and the conversion from rational Bézier to NURPS surfaces.

§1. Basic Concepts

1.1. PS–splines

Let $\Omega \subset \mathbb{R}^2$ be a simply connected subset with polygonal boundary $\delta\Omega$. Let Δ be a conforming triangulation of Ω having n vertices V_i with coordinates (u_i, v_i), $i = 1, \ldots, n$, and let Δ^* be a Powell–Sabin (PS) refinement of Δ (see, e.g. [3]), where each triangle $\rho \in \Delta$ is divided into 6 subtriangles. A Powell–Sabin (PS) spline is a piecewise quadratic polynomial with C^1 continuity on Ω. Dierckx [1] shows how to calculate a normalized B–spline basis for PS–splines:

Definition 1. *A PS–spline surface has a normalized B–spline representation*

$$s(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{3} \mathbf{c_{i,j}} B_i^j(u,v), \qquad (u,v) \in \Omega, \tag{1}$$

where $\mathbf{c_{i,j}} = (c_{i,j}^x, c_{i,j}^y, c_{i,j}^z)$ *are the B–spline control points and* $B_i^j(u,v)$ *are the normalized B–splines.*

This representation shares a number of properties with tensor–product B–splines, making it a powerful tool for representing surfaces in CAGD. We summarize the most important properties here. For details we refer to the original paper [1].

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 45–54.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

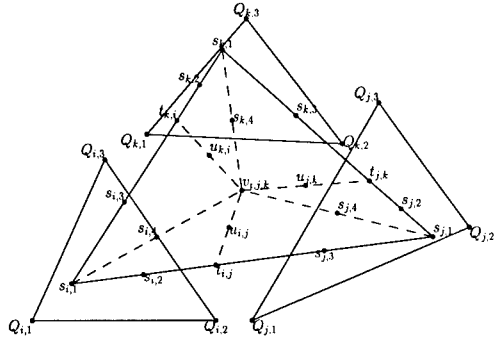


Fig. 1. Domain triangle.

Property 1. $\{B_i^j(u,v)\}_{i=1,\ldots,n}^{j=1,2,3}$ *is a partition of unity:*

$$\begin{cases} B_i^j(u,v) \geq 0, \ (u,v) \in \Omega, \\ \sum_{i=1}^{n} \sum_{j=1}^{3} B_i^j(u,v) \equiv 1, \ (u,v) \in \Omega. \end{cases}$$

Furthermore, $B_{i,j}(u,v)$ is nonzero only on triangles $\rho \in \Delta$ having V_i as a vertex:

Property 2.

$$B_i^j(V_l) = \frac{\partial B_i^j(V_l)}{\partial u} = \frac{\partial B_i^j(V_l)}{\partial v} = 0, \qquad l \neq i. \tag{2}$$

The local control, affine invariance and convex hull properties follow immediately. Linear functions can be represented exactly. In particular, we will make use of the representations

$$u = \sum_{i=1}^{n} \sum_{j=1}^{3} U_{i,j} B_i^j(u,v), \qquad v = \sum_{i=1}^{n} \sum_{j=1}^{3} V_{i,j} B_i^j(u,v)$$

Definition 2. *The PS–triangles* $t_l(Q_{l,1}, Q_{l,2}, Q_{l,3})$, $l = 1, \ldots, n$ *in the planar domain have as vertices the B–spline ordinates* $Q_{l,j}(U_{l,j}, V_{l,j})$, $j = 1, 2, 3$.

Consider a domain triangle $\rho_{i,j,k}(V_i, V_j, V_k) \in \Delta$ with its PS–refinement (see Figure 1). Denote the Bézier ordinates as $s_{v,l}$, $v = i, j, k, l = 1, 2, 3, 4$; $t_{l,m}, u_{l,m}$, $(l,m) \in \{(i,j),(j,k),(k,i)\}$ and $v_{i,j,k}$. They can be written as unique barycentric combinations of the B–spline ordinates:

$$s_{v,l} = \alpha_{v,l} \ Q_{v,1} + \beta_{v,l} \ Q_{v,2} + \gamma_{v,l} \ Q_{v,3}, \tag{3}$$

$$t_{l,m} = \delta_{l,m} \ s_{l,2} + \epsilon_{l,m} \ s_{m,3}, \tag{4}$$

$$u_{l,m} = \delta_{l,m} \ s_{l,4} + \epsilon_{l,m} \ s_{m,4}, \tag{5}$$

$$v_{i,j,k} = \lambda_{i,j,k} \ s_{i,4} + \mu_{i,j,k} \ s_{j,4} + \nu_{i,j,k} \ s_{k,4}. \tag{6}$$

For a given PS–refinement Δ^*, the position of the Bézier ordinates is fixed. This is not the case for the B–spline ordinates. The following lemma however states that there is a restriction on the B–spline ordinates.

Lemma 1. *In order for the basis functions* $\{B_i^j(u,v)\}_{i=1,\ldots,n}^{j=1,2,3}$ *to constitute a partition of unity on* Ω*, it is required that for each vertex* V_i*,* $i = 1,\ldots,n$*, the PS-triangle* $t_i(Q_{i,1}, Q_{i,2}, Q_{i,3})$ *contains the Powell–Sabin points, i.e., the Bézier ordinates* $s_{i,l}, l = 1,2,3,4$*, of any domain triangle having* V_i *as one of its vertices.*

There is a one–one connection between the barycentric coordinates of the Powell–Sabin points at vertex V_i with respect to t_i and the value of the basis functions $B_i^j(u,v)$, $j = 1,2,3$, and of their derivatives at V_i, e.g.

$$B_i^1(V_i) = \alpha_{i,1},\ B_i^2(V_i) = \beta_{i,1},\ B_i^3(V_i) = \gamma_{i,1}. \tag{7}$$

Given a PS–spline surface (1), the corresponding Bézier net can be calculated efficiently by using convex barycentric combinations of the B–spline control points only:

Property 3. *Applying equations (3)–(6) where the ordinates are replaced by control points, yields the corresponding Bézier net of the surface.*

Finally, via the concept of control triangles, the B–spline control points give us valuable insight into the shape of the surface:

Definition 3. *The control triangles are defined as* $T_l(\mathbf{c}_{l,1}, \mathbf{c}_{l,2}, \mathbf{c}_{l,3})$.

Property 4. *Each control triangle* $T_l(\mathbf{c}_{l,1}, \mathbf{c}_{l,2}, \mathbf{c}_{l,3})$ *is tangent to the PS–surface at* $s(V_l)$.

1.2. NURPS

The Normalized B–spline theory for PS–surfaces can now be extended to a rational scheme just like tensor product B–splines are extended to NURBS. Referring to Figure 1, we use the boldface notation for the Bézier points, e.g. $\mathbf{s}_{\mathbf{v},\mathbf{l}}$. Points in homogeneous space get a h–superscript, e.g. $\mathbf{s}^{\mathbf{h}}_{\mathbf{v},\mathbf{l}}$. Their components are $\left(s_{v,l}^{h,x}, s_{v,l}^{h,y}, s_{v,l}^{h,z}, s_{v,l}^{w}\right)$.

Definition 4. *A* Non Uniform Rational Powell–Sabin (NURPS) spline surface *has the form*

$$s(u,v) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{3} \mathbf{c}_{\mathbf{i},\mathbf{j}} w_{i,j} B_i^j(u,v)}{\sum_{i=1}^{n}\sum_{j=1}^{3} w_{i,j} B_i^j(u,v)}, \qquad (u,v) \in \Omega, \tag{8}$$

where $\mathbf{c}_{\mathbf{i},\mathbf{j}} = (c_{i,j}^x, c_{i,j}^y, c_{i,j}^z)$ *are the B–spline control points. We impose that* $w_{i,j} > 0$ *in order for* $s(u,v)$ *to be defined anywhere on* Ω.

If $w_{i,j} = 1$, $i = 1,\ldots,n$, $j = 1,2,3$, then (8) reduces to (1). The following properties are readily verified:

Property 5.

$$s(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{3} \mathbf{c_{i,j}} \phi_i^j(u,v), \tag{9}$$

where

$$\phi_i^j(u,v) = \frac{w_{i,j} B_i^j(u,v)}{\sum_{i=1}^{n} \sum_{j=1}^{3} w_{i,j} B_i^j(u,v)} \tag{10}$$

and

$$\begin{cases} \phi_i^j(u,v) \geq 0, \ (u,v) \in \Omega, \\ \sum_{i=1}^{n} \sum_{j=1}^{3} \phi_i^j(u,v) \equiv 1, \ (u,v) \in \Omega. \end{cases}$$

Furthermore, $\phi_{i,j}(u,v)$ is nonzero only on triangles $\rho \in \Delta$ having V_i as a vertex.

This again implies the local control, affine invariance, and convex hull properties.

Property 6. *A NURPS representation (8) is the 3D–projection in Euclidean space of a 4D PS–spline in homogeneous space:*

$$s(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{3} \mathbf{c^h_{i,j}} B_i^j(u,v), \tag{11}$$

$$\mathbf{c^h_{i,j}} = (w_{i,j} c^x_{i,j}, w_{i,j} c^y_{i,j}, w_{i,j} c^z_{i,j}, w_{i,j}). \tag{12}$$

§2. Evaluation and Subdivision

The evaluation of $s(u,v)$ is performed in two steps:

- First, the corresponding rational piecewise Bézier representation is calculated.
- Then, the rational de Casteljau–algorithm calculates a point on this rational piecewise quadratic Bézier surface. This section shows how to perform the first step in a numerically stable way. For the second step, we refer to Farin [2], Chapter 17.9.

2.1. In homogeneous space

Formulae (3)–(6) can be applied directly in homogeneous space, e.g.

$$\begin{aligned} \mathbf{s^h_{v,l}} &= \alpha_{v,l} \ \mathbf{c^h_{v,1}} + \beta_{v,l} \ \mathbf{c^h_{v,2}} + \gamma_{v,l} \ \mathbf{c^h_{v,3}} \\ &= \left(s^{h,x}_{v,l}, s^{h,y}_{v,l}, s^{h,z}_{v,l}, s^{w}_{v,l} \right) \end{aligned} \tag{13}$$

$$\begin{aligned} \mathbf{v^h_{i,j,k}} &= \lambda_{i,j,k} \ \mathbf{s^h_{i,4}} + \mu_{i,j,k} \ \mathbf{s^h_{j,4}} + \nu_{i,j,k} \ \mathbf{s^h_{k,4}} \\ &= \left(v^{h,x}_{i,j,k}, v^{h,y}_{i,j,k}, v^{h,z}_{i,j,k}, v^{w}_{i,j,k} \right) \end{aligned} \tag{14}$$

Projection back to Euclidean space yields

$$\mathbf{s_{v,l}} = \left(\frac{s^{h,x}_{v,l}}{s^{w}_{v,l}}, \frac{s^{h,y}_{v,l}}{s^{w}_{v,l}}, \frac{s^{h,z}_{v,l}}{s^{w}_{v,l}} \right) \quad \mathbf{v_{i,j,k}} = \left(\frac{v^{h,x}_{i,j,k}}{v^{w}_{i,j,k}}, \frac{v^{h,y}_{i,j,k}}{v^{w}_{i,j,k}}, \frac{v^{h,z}_{i,j,k}}{v^{w}_{i,j,k}} \right).$$

This algorithm has a serious drawback: if the weights vary greatly in magnitude, the coordinates $s^{h,r}_{v,l}, v^{h,r}_{i,j,k}$, $r = x, y, z$ are *blown away*; the calculations don't operate in the convex hull of the control net anymore, and numerical stability is endangered.

2.2. A rational algorithm

The idea behind the rational de Casteljau–algorithm from Farin [2] is to improve numerical stability by rearranging the calculations, avoiding working in homogeneous space:

$$s^{w}_{v,l} = \alpha_{v,l}\ w_{v,1} + \beta_{v,l}\ w_{v,2} + \gamma_{v,l}\ w_{v,3}. \tag{15}$$

Set

$$\tilde{\alpha}_{v,l} = \frac{\alpha_{v,l}\ w_{v,1}}{s^{w}_{v,l}} \geq 0, \quad \tilde{\beta}_{v,l} = \frac{\beta_{v,l}\ w_{v,2}}{s^{w}_{v,l}} \geq 0, \quad \tilde{\gamma}_{v,l} = \frac{\gamma_{v,l}\ w_{v,3}}{s^{w}_{v,l}} \geq 0. \tag{16}$$

Then

$$\mathbf{s_{v,l}} = \tilde{\alpha}_{v,l}\ \mathbf{c_{v,1}} + \tilde{\beta}_{v,l}\ \mathbf{c_{v,2}} + \tilde{\gamma}_{v,l}\ \mathbf{c_{v,3}} \tag{17}$$

with

$$\tilde{\alpha}_{v,l} + \tilde{\beta}_{v,l} + \tilde{\gamma}_{v,l} = 1. \tag{18}$$

The point $\mathbf{s_{v,l}}$ is a convex barycentric combination of $\mathbf{c_{v,1}}, \mathbf{c_{v,2}}$ and $\mathbf{c_{v,3}}$, so numerical stability is guaranteed. Likewise, we find

$$t^{w}_{l,m} = \delta_{l,m}\ s^{w}_{l,2} + \epsilon_{l,m}\ s^{w}_{m,3}, \tag{19}$$

$$u^{w}_{l,m} = \delta_{l,m}\ s^{w}_{l,4} + \epsilon_{l,m}\ s^{w}_{m,4}, \tag{20}$$

$$v^{w}_{i,j,k} = \lambda_{i,j,k}\ s^{w}_{i,4} + \mu_{i,j,k}\ s^{w}_{j,4} + \nu_{i,j,k}\ s^{w}_{k,4}, \tag{21}$$

$$\tilde{\delta}_{l,m} = \frac{\delta_{l,m}\ s^{w}_{l,2}}{t^{w}_{l,m}}, \quad \tilde{\epsilon}_{l,m} = \frac{\epsilon_{l,m}\ s^{w}_{m,3}}{t^{w}_{l,m}}, \tag{22}$$

$$\tilde{\delta}^{*}_{l,m} = \frac{\delta_{l,m}\ s^{w}_{l,4}}{u^{w}_{l,m}}, \quad \tilde{\epsilon}^{*}_{l,m} = \frac{\epsilon_{l,m}\ s^{w}_{m,4}}{u^{w}_{l,m}}, \tag{23}$$

$$\tilde{\lambda}_{i,j,k} = \frac{\lambda_{i,j,k}\ s^{w}_{i,4}}{v^{w}_{i,j,k}}, \quad \tilde{\mu}_{i,j,k} = \frac{\mu_{i,j,k}\ s^{w}_{j,4}}{v^{w}_{i,j,k}}, \quad \tilde{\nu}_{i,j,k} = \frac{\nu_{i,j,k}\ s^{w}_{k,4}}{v^{w}_{i,j,k}}, \tag{24}$$

where

$$\tilde{\delta}_{l,m} + \tilde{\epsilon}_{l,m} = \tilde{\delta}^{*}_{l,m} + \tilde{\epsilon}^{*}_{l,m} = \tilde{\lambda}_{i,j,k} + \tilde{\mu}_{i,j,k} + \tilde{\nu}_{i,j,k} = 1,$$

and finally

$$\mathbf{t_{l,m}} = \tilde{\delta}_{l,m}\ \mathbf{s_{l,2}} + \tilde{\epsilon}_{l,m}\ \mathbf{s_{m,3}}, \tag{25}$$

$$\mathbf{u_{l,m}} = \tilde{\delta}^*_{l,m}\ \mathbf{s_{l,4}} + \tilde{\epsilon}^*_{l,m}\ \mathbf{s_{m,4}}, \tag{26}$$

$$\mathbf{v_{i,j,k}} = \tilde{\lambda}_{i,j,k}\ \mathbf{s_{i,4}} + \tilde{\mu}_{i,j,k}\ \mathbf{s_{j,4}} + \tilde{\nu}_{i,j,k}\ \mathbf{s_{k,4}}. \tag{27}$$

All formulae are convex barycentric combinations operating in the convex hull of the B–spline control net. After having computed the rational Bézier representation, Farin's rational de Casteljau algorithm can be used to evaluate the surface at any point $(u, v) \in \Omega$.

2.3. Subdivision on uniform triangulations

The evaluation and subdivision of spline curves and surfaces are closely related problems. For the particular case of a uniform triangulation Δ, a subdivision scheme for PS–surfaces has been derived [4]. As an application, it was shown how a wireframe of the surface can be calculated in an efficient and numerically stable way. This scheme can easily be extended to NURPS on uniform triangulations again using Farin's technique from the previous section. The details are omitted here.

§3. Control Planes

Recall that the NURPS representation inherits the convex hull, affine invariance, and local control property from the normalized B–spline representation. This section adds the tangent property to the inheritance list, and shows how the rational representation allows for more flexibilty when designing surfaces.

3.1. Tangent property

Referring to the locality of the B–splines (2), it is easy to verify that the evaluation of $s(u, v)$ and its derivatives at vertex V_i yields

$$s(V_i) = \tilde{\alpha}_{i,1}\ \mathbf{c_{i,1}} + \tilde{\beta}_{i,1}\ \mathbf{c_{i,2}} + \tilde{\gamma}_{i,1}\ \mathbf{c_{i,3}}, \tag{28}$$

$$\frac{\partial s(V_i)}{\partial u} = e_{i,1}\ \mathbf{c_{i,1}} + e_{i,2}\ \mathbf{c_{i,2}} + e_{i,3}\ \mathbf{c_{i,3}}, \tag{29}$$

$$\frac{\partial s(V_i)}{\partial v} = d_{i,1}\ \mathbf{c_{i,1}} + d_{i,2}\ \mathbf{c_{i,2}} + d_{i,3}\ \mathbf{c_{i,3}}, \tag{30}$$

for some

$$e_{i,1} + e_{i,2} + e_{i,3} = d_{i,1} + d_{i,2} + d_{i,3} = 0.$$

It follows that the control triangle at V_i is tangent to the surface at $s(V_i)$, i.e., any point $\mathbf{p}$ in the tangent plane is a barycentric combination of the control points $\mathbf{c_{i,1}}, \mathbf{c_{i,2}}, \mathbf{c_{i,3}}$:

$$\mathbf{p} = s(V_i) + a\ \frac{\partial s(V_i)}{\partial u} + b\ \frac{\partial s(V_i)}{\partial v},\ a, b \in \mathbb{R}.$$

This is illustrated in Figure 2 (left).

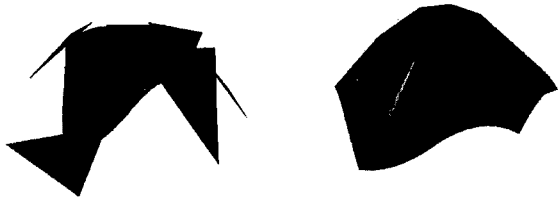


Fig. 2. NURPS surface and its control planes; local planar effects.

3.2. Shape parameters

Farin [2] introduces the concept of shape parameters with respect to rational Bézier curves. A geometric handle allows the designer to influence the shape of the curve in a predictable way, rather than requiring the input of numbers for the weights. In the same work, it is stated that this property does not carry over to rational Bézier surfaces on triangles, but shape parameters can be defined for NURPS.

Recall that $(\tilde{\alpha}_{v,1}, \tilde{\beta}_{v,1}, \tilde{\gamma}_{v,1})$ are the barycentric coordinates of $\mathbf{s_{v,1}}$ with respect to control triangle $T_v(\mathbf{c_{v,1}}, \mathbf{c_{v,2}}, \mathbf{c_{v,3}})$. From (16) it follows that $\mathbf{s_{v,1}}$ can be moved within T_v to a new location $(\tilde{\alpha}'_{v,1}, \tilde{\beta}'_{v,1}, \tilde{\gamma}'_{v,1})$, while keeping its weight $s^w_{v,1}$ constant. The corresponding PS–weights are found immediately as

$$w_{v,1} = \frac{\tilde{\alpha}'_{v,1} s^w_{v,1}}{\alpha_{v,1}}, \quad w_{v,2} = \frac{\tilde{\beta}'_{v,1} s^w_{v,1}}{\beta_{v,1}}, \quad w_{v,3} = \frac{\tilde{\gamma}'_{v,1} s^w_{v,1}}{\gamma_{v,1}} \tag{31}$$

This shows how $(\tilde{\alpha}_{v,1}, \tilde{\beta}_{v,1}, \tilde{\gamma}_{v,1})$ can be used as shape parameters.

3.3. Planar sections

Definition 5. *Let* $[\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}]$ *denote the* convex hull *of the 3D points* $\mathbf{t_1}, \mathbf{t_2}, \ldots, \mathbf{t_n}$.

Definition 6. *Let* $S(A)$ *denote the image of a subset* $A \subset \Omega$ *under (8).*

Definition 7. *Let* $\tau(a, b, c)$ *denote the* Bézier subtriangle *in the domain plane with vertices* a, b *and* c.

If the control triangles of adjacent vertices V_i, V_j, V_k are chosen to be coplanar, then the surface section $S(\rho_{i,j,k}(V_i, V_j, V_k))$ will be in the same plane, as a consequence of the convex hull property. However, using the weights in the NURPS representation, it is possible to achieve more local planar effects.

The rational evaluation algorithm from Section 2.2 reveals that for $w_{i,1} = w_{i,2} = w_{i,3} = w \to \infty$, $i \in (1, \ldots, n)$ and referring to Figure 1, the following holds on the domain triangle $\rho_{i,j,k}(V_i, V_j, V_k)$:

$$\mathbf{t_{i,j}} = \frac{\delta_{i,j}}{\delta_{i,j} + \frac{\epsilon_{i,j} s^w_{j,3}}{w}} \mathbf{s_{i,2}} + \frac{\frac{\epsilon_{i,j} s^w_{j,3}}{w}}{\delta_{i,j} + \frac{\epsilon_{i,j} s^w_{j,3}}{w}} \mathbf{s_{j,3}}. \tag{32}$$

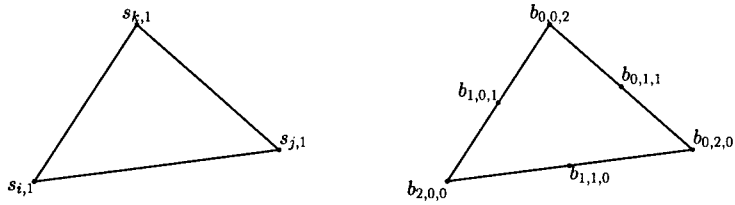


Fig. 3. Bézier triangle.

Thus,

$$\lim_{w\to\infty} \mathbf{t_{i,j}} = \mathbf{s_{i,2}}. \tag{33}$$

Likewise, for the other Bézier points of $\tau\,(s_{i,1}, t_{i,j}, v_{i,j,k})$, we find

$$\mathbf{s_{i,l}} = \alpha_{i,l}\ \mathbf{c_{i,1}} + \beta_{i,l}\ \mathbf{c_{i,2}} + \gamma_{i,l}\ \mathbf{c_{i,3}},\ \ l = 1, 2, 4, \tag{34}$$

$$\lim_{w\to\infty} \mathbf{u_{i,j}} = \lim_{w\to\infty} \mathbf{v_{i,j,k}} = \mathbf{s_{i,4}}. \tag{35}$$

Consequently,

$$S\,(\tau\,(s_{i,1}, t_{i,j}, v_{i,j,k})) = [\mathbf{s_{i,1}}, \mathbf{s_{i,2}}, \mathbf{s_{i,4}}]\,.$$

Similar reasoning on the other Bézier subtriangles shows that

$$\begin{aligned}
S\,(\tau\,(s_{i,1}, v_{i,j,k}, t_{k,i})) &= [\mathbf{s_{i,1}}, \mathbf{s_{i,4}}, \mathbf{s_{i,3}}]\,,\\
S\,(\tau\,(t_{i,j}, s_{j,1}, v_{i,j,k})) &= [\mathbf{s_{i,2}}, \mathbf{s_{i,4}}]\,,\\
S\,(\tau\,(t_{k,i}, v_{i,j,k}, s_{k,1})) &= [\mathbf{s_{i,3}}, \mathbf{s_{i,4}}]\,,\\
S\,(\tau\,(v_{i,j,k}, s_{j,1}, t_{j,k})) &= [\mathbf{s_{i,4}}]\,,\\
S\,(\tau\,(v_{i,j,k}, t_{j,k}, s_{k,1})) &= [\mathbf{s_{i,4}}]\,,
\end{aligned}$$

and therefore,

$$S\,(\rho_{i,j,k}\,(V_i, V_j, V_k)) = [\mathbf{s_{i,1}}, \mathbf{s_{i,2}}, \mathbf{s_{i,4}}, \mathbf{s_{i,3}}] \subset [\mathbf{c_{i,1}}, \mathbf{c_{i,2}}, \mathbf{c_{i,3}}]$$

The latter image is a planar surface section. Figure 2 (right) shows some NURPS surface with very large weights at a vertex.

§4. Conversion from Rational Bézier to NURPS Representation

Suppose we are given a rational quadratic Bézier surface on one domain triangle (see Figure 3) $\rho(s_{i,1}, s_{j,1}, s_{k,1})$

$$b(u,v) = \sum_{i_1+i_2+i_3=2} \mathbf{b}^h_{i_1,i_2,i_3} B^2_{i_1,i_2,i_3}(t_1,t_2,t_3), \tag{36}$$

where $i_1, i_2, i_3 > 0, (u,v) \in \rho$ and (t_1,t_2,t_3) are the barycentric coordinates of (u,v) with respect to ρ. In this section it is shown how a NURPS representation

$$s(u,v) = \sum_{l=i,j,k} \sum_{m=1}^{3} \mathbf{c}^h_{l,m} B^m_l(u,v) \tag{37}$$

of the given surface, for a specific choice of the PS-triangles, is immediately obtained. To simplify the notation, the surfaces are considered in homogeneous space.

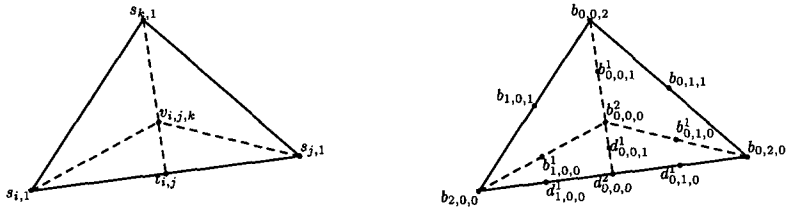


Fig. 4. Subdivision at $v_{i,j,k}$ and $t_{i,j}$.

Lemma 2. *Suppose we are given a triangle* $t(V_i, V_j, V_k)$ *with barycenter* z. *If* W_l *denotes the midpoint of the side opposite to* V_l, *then* $(z + V_l)/2$ *is the barycenter of the triangle* $t(V_l, W_m, W_n)$, $l, m, n \in \{i, j, k\}$, $l \neq m \neq n$.

The construction of the NURPS representation relies on the de Casteljau-algorithm for Bézier triangles (see, e.g., [2]). Subdivision at the barycenter of ρ and at the midpoint of edge $s_{i,1}s_{j,1}$ (see Figure 4) yields the new Bézier points

$$\mathbf{b}^{1,h}_{1,0,0} = \frac{1}{3}\left(\mathbf{b}^h_{2,0,0} + \mathbf{b}^h_{1,1,0} + \mathbf{b}^h_{1,0,1}\right), \tag{38}$$

$$\mathbf{b}^{1,h}_{0,1,0} = \frac{1}{3}\left(\mathbf{b}^h_{1,1,0} + \mathbf{b}^h_{0,2,0} + \mathbf{b}^h_{0,1,1}\right), \tag{39}$$

$$\mathbf{b}^{1,h}_{0,0,1} = \frac{1}{3}\left(\mathbf{b}^h_{1,0,1} + \mathbf{b}^h_{0,1,1} + \mathbf{b}^h_{0,0,2}\right), \tag{40}$$

$$\mathbf{b}^2_{0,0,0} = \frac{1}{3}\left(\mathbf{b}^{1,h}_{1,0,0} + \mathbf{b}^{1,h}_{0,1,0} + \mathbf{b}^{1,h}_{0,0,1}\right), \tag{41}$$

$$\mathbf{d}^{1,h}_{1,0,0} = \frac{1}{2}(\mathbf{b}^h_{2,0,0} + \mathbf{b}^h_{1,1,0}), \tag{42}$$

$$\mathbf{d}^{1,h}_{0,1,0} = \frac{1}{2}(\mathbf{b}^h_{1,1,0} + \mathbf{b}^h_{0,2,0}), \tag{43}$$

$$\mathbf{d}^{1,h}_{0,0,1} = \frac{1}{2}(\mathbf{b}^{1,h}_{1,0,0} + \mathbf{b}^{1,h}_{0,1,0}), \tag{44}$$

$$\mathbf{d}^{2,h}_{0,0,0} = \frac{1}{2}(\mathbf{d}^{1,h}_{1,0,0} + \mathbf{d}^{1,h}_{0,1,0}). \tag{45}$$

After subdivision of the two remaining edges, the 6 subtriangles thus obtained constitute a PS–refinement of ρ, say, with interior point $v_{i,j,k}$ and edge points $t_{i,j}, t_{j,k}, t_{k,i}$ (see Figure 5, left). A NURPS representation of the given surface on this PS–refinement is easily obtained (see Figure 5, right). Set

$$Q_{v,1} = s_{v,1},\ v = i, j, k,$$
$$Q_{i,2} = Q_{j,3} = t_{i,j}, \quad Q_{j,2} = Q_{k,3} = t_{j,k}, \quad Q_{k,2} = Q_{i,3} = t_{k,i},$$

and

$$\mathbf{c}^h_{i,1} = \mathbf{b}^h_{2,0,0},\ \mathbf{c}^h_{i,2} = \mathbf{b}^h_{1,1,0},\ \mathbf{c}^h_{i,3} = \mathbf{b}^h_{1,0,1},$$
$$\mathbf{c}^h_{j,1} = \mathbf{b}^h_{0,2,0},\ \mathbf{c}^h_{j,2} = \mathbf{b}^h_{0,1,1},\ \mathbf{c}^h_{j,3} = \mathbf{b}^h_{1,1,0},$$
$$\mathbf{c}^h_{k,1} = \mathbf{b}^h_{0,0,2},\ \mathbf{c}^h_{k,2} = \mathbf{b}^h_{1,0,1},\ \mathbf{c}^h_{k,3} = \mathbf{b}^h_{0,1,1}.$$

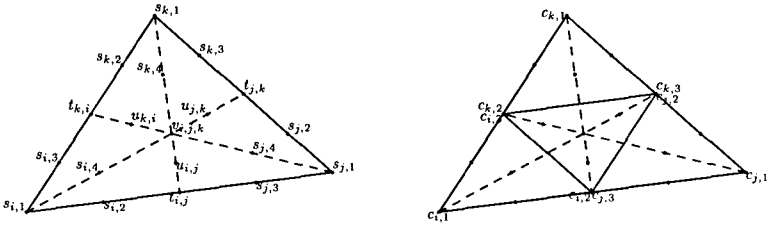


Fig. 5. PS–refinement.

Then by Lemma 2, it follows that the PS–points $s_{v,l}$, $l = 1, 2, 3, 4$ are inside the PS–triangle t_v, $v = i, j, k$. Now recall formula (13) for $v = i, j, k$, and $l = 4$, and formula (14), with $\alpha_{v,4} = \beta_{v,4} = \gamma_{v,4} = \frac{1}{3}$, resp. $\lambda_{i,j,k} = \mu_{i,j,k} = \nu_{i,j,k} = \frac{1}{3}$, in order to calculate the corresponding Bézier points of this NURPS surface. It turns out that these equations are exactly the same as the subdivision formulae (38)–(41). Likewise, since in (3)–(5)

$$(\alpha_{v,2}, \beta_{v,2}, \gamma_{v,2}) = (\frac{1}{2}, \frac{1}{2}, 0),$$
$$(\alpha_{v,3}, \beta_{v,3}, \gamma_{v,3}) = (\frac{1}{2}, 0, \frac{1}{2}),$$
$$(\delta_{l,m}, \epsilon_{l,m}) = (\frac{1}{2}, \frac{1}{2}),$$

for $v = i, j, k$ and $(l, m) \in \{(i, j), (j, k), (k, i)\}$, similar reasoning shows that calculating the corresponding Bézier net of (37) exactly yields the Bézier net of (36) after the proposed subdivisions. Hence, $b(u, v) \equiv s(u, v)$ on ρ.

References

1. Dierckx, P., On calculating normalized Powell–Sabin B–splines, Comput. Aided Geom. Design **15** (1997), 61–78.
2. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*, 4th edition, Academic Press, Boston, 1997.
3. Powell, M. J. D. and M. A. Sabin, Piecewise quadratic approximations on triangles, ACM Trans. Math. Software **3** (1977), 316–325.
4. Windmolders, J., and P. Dierckx, Subdivision of uniform Powell–Sabin splines, Comput. Aided Geom. Design **16** (1999), 301–315.

Paul Dierckx and Joris Windmolders
Celestijnenlaan 200A
3001 Heverlee
Belgium
paul.dierckx@cs.kuleuven.ac.be
joris.windmolders@cs.kuleuven.ac.be

Distance Calculation Between a Point and a NURBS Surface

Eva Dyllong and Wolfram Luther

Abstract. In this paper, we consider the computation of an Euclidean shortest path between a point and a modelled curve or surface in three-dimensional space, which is one of the fundamental problems in robotics and many other areas. A new accurate algorithm for the distance-calculation between a point and a NURBS curve and its extension to the case of a point and a NURBS surface is presented. The algorithm consists of two steps, and is crucially based on appropriate projections and subdivision techniques. To solve a nonlinear polynomial system derived from the classical formulation of the distance problem, the well-known Newton-type algorithms or subdivision-based techniques first considered by Sherbrooke and Patrikalakis are used. Their modifications in conjunction with a low subdivision depth in the presented algorithms yield a verified enclosure of the solution.

§1. Introduction

The distance-calculation is an essential component of robot motion planning and control to steer the robot away from its surrounding obstacles or to work on a target surface. The obstacles may be polyhedral objects, quadratic surfaces, which include spherical and cylindrical surfaces or more general surface types like the non-uniform rational B-splines (NURBS). Most of the well-known algorithms in the fields of computational geometry, robotics and Computer Aided Design are focused on computing the distance between polyhedra, as the problem is easier to solve and the answer is sufficient for many problems. For example, if a *free-form* designed obstacle like a NURBS surface is located at a great distance from a moving robot, then it is sufficient in the next step to know the distance values from certain sensor points on the robot to the convex hull of the NURBS control points, which describes a convex polyhedron, instead of the more time-consuming and expensive computation of the exact distance values. But if the robot approaches an obstacle, more details are necessary, and fast and accurate algorithms for finding the nearest point on the NURBS curve or surface are highly recommended.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 55–62.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

There is an abundance of literature to calculate the distance between convex and non-convex objects. For convex polyhedra, a lot of algorithms exist [1,4]. Two well-known algorithms, the Gilbert method (GJK, [8]) and the algorithm of Canny and Lin (CL, [3]) present iterative solutions to the problem, i.e., both construct a sequence of pairs of proposed distance points which are then improved by gradient descent. Another wide field consists of algorithms for general, mainly convex objects [5,7,13]. But in particular for objects defined by NURBS, there are only a few contributions in the literature. In [2], Cameron and Turnbull focus on computing the distance between convex objects defined by NURBS curves or patches, for which the critical step is the evaluation of the support mapping. The method is based on the Gilbert-Foo algorithm [7] for general convex objects with adjustments of the termination criteria like the support mapping for NURBS, which uses derivatives of their basis functions and the Newton-Raphson method (NR solver) for finding the roots. An algorithm for the computation of stationary points of a squared distance function is presented in [11]. This problem is converted to n_e polynomial equations with n_e variables expressed in a tensor product Bernstein basis. The solution method uses subdivision relying on the convex hull property of Bernstein polynomials and minimization techniques.

In this paper we describe a new algorithm which consists of two steps, and is mainly based on accurate projections and subdivision techniques. In the first step, the NURBS curve is decomposed into rational Bézier segments. Then some evaluations of suitable scalar products decide on further subdivision of a rational Bézier segment. This subdivision is iterated until certain criteria are fulfilled. In addition, a composition of the new method together with the classical formulation of the distance problem based on the calculation of a solution of nonlinear polynomial systems is presented. The algorithm is extended to the case of a NURBS surface.

§2. Problem Formulation

A NURBS curve $\vec{C}(u)$ of degree p is a vector-valued function of one parameter defined by

$$\vec{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u)\omega_i \vec{P}_i}{\sum_{i=0}^{n} N_{i,p}(u)\omega_i}, \qquad a \le u \le b,$$

where $\{\vec{P}_i = (x_i, y_i, z_i) \in \mathbb{R}^3\}_{i=0}^{n}$ are the control points, $\{\omega_i\}_{i=0}^{n}$ are the weights, and $\{N_{i,p}(u)\}_{i=0}^{n}$ are the pth-degree B-spline basis functions defined on the knot sequence $U = \{u_i\}_{i=0}^{n+p+1}$ with $\{u_i = a\}_{i=0}^{p}$ and $\{u_i = b\}_{i=n+1}^{n+p+1}$. Let $\vec{P}^\omega = (\omega x, \omega y, \omega z, \omega) = (x', y', z', \omega') \in \mathbb{R}^4$ and H be the perspective map given by

$$\vec{P} = H\{\vec{P}^\omega\} = H\{(x', y', z', \omega')\} = \begin{cases} (\frac{x'}{\omega'}, \frac{y'}{\omega'}, \frac{z'}{\omega'}), & \text{if } \omega' \neq 0, \\ (x', y', z'), & \text{if } \omega' = 0. \end{cases}$$

Applying H to the nonrational B-spline curve in homogeneous coordinates

$$\vec{C}^\omega(u) = \sum_{i=0}^{n} N_{i,p}(u)\vec{P}_i^\omega \tag{1}$$

yields the corresponding NURBS curve $\vec{C}(u)$, i.e., $\vec{C}(u) = H\{\vec{C}^{\omega}(u)\}$.

Similarly to the curve case, we define a NURBS surface using the tensor product scheme. Accordingly, a NURBS surface $\vec{S}(u,v)$ is a bivariate vector-valued piecewise rational function of the form

$$\vec{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\omega_{i,j}\vec{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\omega_{i,j}}, \qquad a \le u,v \le b,$$

with the bidirectional control net $\vec{P}_{i,j}$, the weights $\omega_{i,j}$ and the B-spline basis functions $N_{i,p}(u)$ and $N_{j,q}(v)$.

For a given point $\vec{Q} \in \mathbb{R}^3$ we address the problem of finding a shortest straight line segment $[\vec{Q},\vec{D}]$ with $\vec{D} \in \vec{C}(u)$ or $\vec{D} \in \vec{S}(u,v)$, respectively. We assume that all weights of the rational curves and surfaces are positive, to ensure that the convex hull property holds.

§3. Distance Algorithm for a Point and a NURBS Curve

In this section, an efficient and accurate algorithm for distance calculation between a given point $\vec{Q}$ and a NURBS curve $\vec{C}(u)$ is presented, which is a kind of an adaptive system of solution methods. The extension to the case of a NURBS surface works analogously.

The algorithm consists of two steps. In the first step, the NURBS curve is decomposed into rational Bézier segments $\vec{C}_j(u)$, $j = 1,\ldots,n_p$, which can be realized once in the preparation phase. In [12] Piegl and Tiller present an efficient algorithm for computing the n_p Bézier segments using the homogeneous form given by (1). Thus, our task is to compute the distance between a point and a rational Bézier curve. After decomposition, the distances between $\vec{Q}$ and each endpoint of the rational Bézier segments $\vec{C}_j(u)$ are calculated, and the smallest value is stored as a first rough approximation to the distance value d. In the second step, the rational Bézier segments are processed gradually. Let $\vec{P}_{j,k}$, $k = 0,\ldots,p$, be the control points of the j-th Bézier segment $\vec{C}_j(u)$, $j \in \{1,\ldots,n_p\}$. Then, for each $\vec{P}_{j,k}$, $k = 1,\ldots,p-1$, the distance to the straight line supporting the line segment $l(\vec{P}_{j,0},\vec{P}_{j,p})$ between the endpoints $\vec{P}_{j,0}$ and $\vec{P}_{j,p}$ is calculated, and for each projection point $\vec{R}_{j,k}$, $k = 1,\ldots,p-1$, on the line, we test if $\vec{R}_{j,k}$ belongs to the line segment $l(\vec{P}_{j,0},\vec{P}_{j,p})$, using suitable scalar product evaluations. If $\vec{R}_{j,k} \notin l(\vec{P}_{j,0},\vec{P}_{j,p})$ for at least one $k \in \{1,\ldots,p-1\}$, then the Bézier segment is subdivided into two Bézier segments, for which the second step of the algorithm has to be started again. Otherwise, the algorithm tests whether the Bézier segment can be replaced by the line segment $l(\vec{P}_{j,0},\vec{P}_{j,p})$ using the theorem by Wang and Xu (see Sec. 3.2). If $\vec{C}_j(u)$ is nearly a straight line, with a given accuracy ε, then the distance between the point $\vec{Q}$ and the line segment $l(\vec{P}_{j,0},\vec{P}_{j,p})$ is calculated. The distance d is updated if a smaller value is found, and j is replaced by $j+1$. If $\vec{C}_j(u)$ is not nearly a line segment, the following scalar products

$$s_k := (\vec{P}_{j,1} - \vec{R}_{j,1}) \cdot (\vec{P}_{j,k} - \vec{R}_{j,1}) \quad \text{for} \quad k = 2,\ldots,p-1,$$

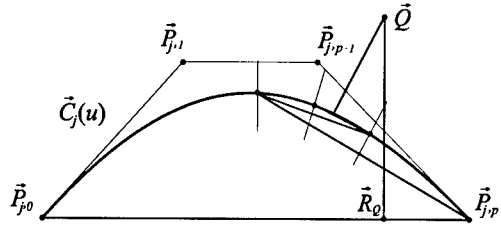


Fig. 1. Convergence of the algorithm.

are calculated. If all $s_k \geq 0$, i.e., all $\vec{P}_{j,k}$, $k = 1, \ldots, p-1$, are on the same side of the line supporting $l(\vec{P}_{j,0}, \vec{P}_{j,p})$, then the algorithm tests the position of point $\vec{Q}$. Otherwise, the Bézier segment $\vec{C}_j(u)$ is subdivided into two segments. To test whether $\vec{Q}$ lies in an influence area of the Bézier segment $\vec{C}_j(u)$, the projection point $\vec{R}_Q$ of $\vec{Q}$ on the line supporting $l(\vec{P}_{j,0}, \vec{P}_{j,p})$ is calculated and its position on the line is checked. If $\vec{R}_Q \notin l(\vec{P}_{j,0}, \vec{P}_{j,p})$, the distances $d(\vec{Q}, \vec{P}_{j,0})$ and $d(\vec{Q}, \vec{P}_{j,p})$ are calculated, d is updated if necessary and j is replaced by $j+1$. Otherwise, the Bézier segment has to be subdivided into two segments to increase the accuracy of the result. Fig. 1 shows how the algorithm works.

The subdivision of a rational Bézier segment can be continued until the termination criteria (see Sec. 3.2) are fulfilled or it can be interrupted after some steps, and afterwards the distance problem can be transformed in terms of the solution of the polynomial equation

$$(\vec{Q} - \vec{C}(u)) \cdot \vec{C}'(u) = 0 \tag{2}$$

in the variable u, where $\vec{C}'(u)$ describes the derivative of the curve $\vec{C}(u)$. This is mainly recommended for NURBS surfaces with a large curvature to avoid a high depth of subdivision. We calculate the roots of this equation using either the well-known (interval) Newton method, or one of the recently implemented solution methods briefly described in Sec. 3.3 (see [10]). A diagram illustrating the outline of the distance algorithms is given in Fig. 2.

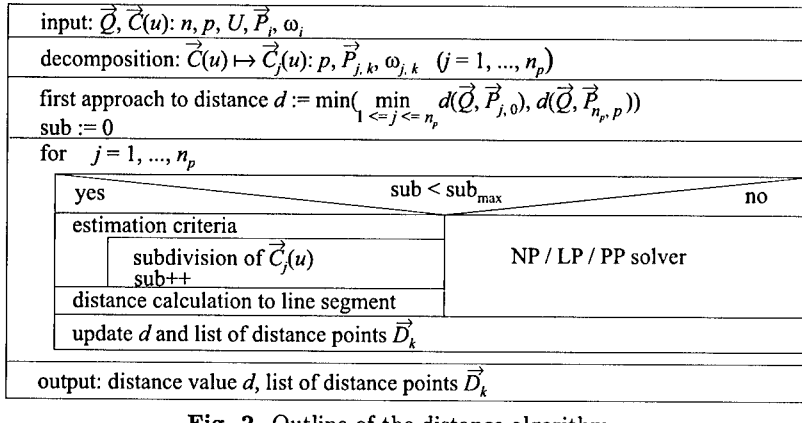


Fig. 2. Outline of the distance algorithm.

3.1. Subdivision

The subdivision of the control polygon of $\vec{C}_j^{\omega}(u)$ determinated by $\{\vec{P}_{j,k}^{\omega}\}_{k=0}^{p}$ into two Bézier segments with control points $\{\vec{Q}_{0,l}^{\omega}\}_{l=0}^{p}$ and $\{\vec{Q}_{1,l}^{\omega}\}_{l=0}^{p}$ works in homogeneous coordinates, and reads as follows:

```
for   k = 0 to p do
begin Q^ω_{0,k} := P^ω_{j,0}; Q^ω_{1,p-k} := P^ω_{j,p-k};
      for l = 0 to p-k-1 do
      P^ω_{j,l} := (P^ω_{j,l} + P^ω_{j,l+1})/2.0   end
```

Extending the idea from Bézier curve $\vec{C}_j^{\omega}(u)$ to Bézier surface $\vec{S}_{i,j}^{\omega}(u,v)$ by calling the routine twice, first in u and then in v direction, a subdivision of $\vec{S}_{i,j}^{\omega}(u,v)$ into four Bézier patches is realized (see [12]).

3.2. Termination criteria

In [15] Wang and Xu prove the following theorem:

Theorem 1. *For the rational Bézier curve $\vec{C}(u)$ of degree p,*

$$d(\vec{C}(u), l(\vec{P}_0, \vec{P}_p)) \leq \Phi(\omega_0, \ldots, \omega_p) \cdot \max_{1 \leq i \leq p-1} d(\vec{P}_i, l(\vec{P}_0, \vec{P}_p)),$$

where $\vec{C}(u) = \sum_{i=0}^{p} B_{i,p}(u)\omega_i \vec{P}_i / \sum_{i=0}^{p} B_{i,p}(u)\omega_i$ with the Bernstein polynomials $B_{i,p}(u)$ of degree p, and

$$\Phi(\omega_0, \ldots, \omega_p) := 1 - \left(1 + \max(\omega_0^{-1}, \omega_p^{-1})(\max_{1 \leq i \leq p-1} \omega_i)(2^{p-1} - 1)\right)^{-1},$$

$$d(\vec{C}(u), l(\vec{P}_0, \vec{P}_p)) := \sup_{a \leq u \leq b} \{\inf_{0 \leq t \leq 1} d(\vec{C}(u), t\vec{P}_0 + (1-t)\vec{P}_p)\}.$$

If after some subdivision steps, $d(\vec{Q}, l(\vec{P}_{j,0}, \vec{P}_{j,p})) \geq d + d(\vec{C}_j(u), l(\vec{P}_{j,0}, \vec{P}_{j,p}))$, or the curve can be approximated by $l(\vec{P}_{j,0}, \vec{P}_{j,p})$, i.e., $d(\vec{C}_j(u), l(\vec{P}_{j,0}, \vec{P}_{j,p}))$ is not greater than the desired tolerance ε, the subdivision of the segment stops after this step.

If $\omega_i = const$, the curve $\vec{C}_j(u)$ describes a Bézier curve, and the termination criterion of Theorem 1 is reduced to testing the following conditions: $(1 - 1/2^{p-1}) d(\vec{P}_{j,k}, l(\vec{P}_{j,0}, \vec{P}_{j,p})) < \varepsilon$ for $k = 1, \ldots, p-1$. In addition, for a Bézier curve the following theorem proved in [14] specifies the number of necessary subdivisions, i.e., after r subdivision steps the curve can be replaced by the line segments:

Theorem 2. *For the Bézier curve $\vec{C}(u)$ of degree p with control points $\{\vec{P}_i = (x_i, y_i, z_i)\}_{i=0}^{p}$ and any given $\varepsilon \geq 0$, let $L := \max_{0 \leq i \leq p-2}\{|x_i - 2x_{i+1} + x_{i+2}|, |y_i - 2y_{i+1} + y_{i+2}|, |z_i - 2z_{i+1} + z_{i+2}|\}$, and $r := \log_4(\sqrt{3}p(p-1)L/(8\varepsilon))$. Then, for $a \leq \alpha < \beta \leq b$ and $-\log_2((\beta - \alpha)/(b - a)) \geq r$,*

$$d(\vec{C}(u), l(\vec{C}(\alpha), \vec{C}(\beta))) \leq \varepsilon.$$

If at the beginning of the subdivision the value r can be calculated, then it is not necessary to perform the termination tests of Theorem 1.

3.3 Solution techniques and complexity analysis

The distance problem can be converted into a problem of computing all roots of a system of nonlinear polynomial equations in one or two variables. There are two techniques designed to solve such a problem in n_e variables efficiently; the projected-polyhedron (PP) and the linear programming (LP) technique, developed by Sherbrooke and Patrikalakis [10]. They rely on representation of polynomials in the multivariate Bernstein basis, the convex hull property and on the subdivision or linear programming technique. Alternatively, the Newton-Raphson method can be used to find the roots of the nonlinear polynomial equations.

Next, we analyse the amount of time required to execute each step of the distance algorithm in case of the NURBS curve. The decomposition of the curve $\vec{C}(u)$ of degree p into n_p Bézier segments takes at most $O(p \cdot n_p)$ operations, and the first approach to the distance value needs n_p+1 steps. The total cost of the distance calculation for n_p Bézier segments with subdivision depth of k is in worst case $O(2^k p^2 \cdot n_p)$ independently of the method used (subdivision-based technique or PP/LP solver) for finding the distance points. In this case ($n_e = 1$), close to a simple root, quadratic convergence is achieved.

§4. Distance Algorithms for a Point and a NURBS Surface

In the case of a NURBS surface $\vec{S}(u,v)$, the distance algorithm maintains its structure. After the decomposition of the surface into $n_p \cdot n_q$ Bézier patches $\vec{S}_{i,j}(u,v)$ with control points $\vec{P}^{i,j}_{k,l}$ ($0 \le k \le p$, $0 \le l \le q$), the first approximation to the distance value d is calculated taking the minimum of distances $d(\vec{Q}, \vec{P}^{i,j}_{k,l})$ for $k = 0, p$ and $l = 0, q$. The termination critera for subdivision of a non-degenerate Bézier patch $\vec{S}_{i,j}(u,v)$ ($\vec{P}^{i,j}_{0,0}$, $\vec{P}^{i,j}_{p,0}$, $\vec{P}^{i,j}_{0,q}$ are not collinear) are modified in the following way:

$$\Phi(\omega_{e,0},.,\omega_{e,q}) d(\vec{P}^{i,j}_{e,l}, l(\vec{P}^{i,j}_{e,0}, \vec{P}^{i,j}_{e,q})) \le \varepsilon,\ \Phi(\omega_{0,l},.,\omega_{p,l}) d(\vec{P}^{i,j}_{k,l}, l(\vec{P}^{i,j}_{0,l}, \vec{P}^{i,j}_{p,l})) \le \varepsilon$$
$$\text{and} \quad (\vec{P}^{i,j}_{p,q} - \vec{P}^{i,j}_{0,0}) \cdot \left((\vec{P}^{i,j}_{p,0} - \vec{P}^{i,j}_{0,0}) \times (\vec{P}^{i,j}_{0,q} - \vec{P}^{i,j}_{0,0}) \right) \le \varepsilon$$

for all $1 \le k \le p$, $1 \le l \le q$, and $e = 0, p$ (where $\times$ denotes the cross product). In this case $\vec{S}_{i,j}(u,v)$ can be replaced by a plane segment defined by $\vec{P}^{i,j}_{0,0}$, $\vec{P}^{i,j}_{p,0}$, and $\vec{P}^{i,j}_{0,q}$, the subdivision of $\vec{S}_{i,j}(u,v)$ is stopped, and the distance between the plane segment and the point $\vec{Q}$ is calculated. The remaining tests are performed in the u and v directions analogously to the curve case using the tensor-product structure of $\vec{S}_{i,j}(u,v)$. If the point $\vec{Q}$ does not lie in the influence area of the Bézier patch $\vec{S}_{i,j}(u,v)$, i.e., the projections onto the lines forming the boundary of the triangle defined by $\vec{P}^{i,j}_{0,0}$, $\vec{P}^{i,j}_{p,0}$, and $\vec{P}^{i,j}_{0,q}$ or defined by $\vec{P}^{i,j}_{p,q}$, $\vec{P}^{i,j}_{p,0}$ and $\vec{P}^{i,j}_{0,q}$ do not belong to sides of the triangle, the distance between the point $\vec{Q}$ and one of the boundary lines $\vec{S}_{i,j}(u,v)$, $u \in \{a,b\}$, $v \in \{c,d\}$, is calculated, d is updated if necessary, and the subdivision of $\vec{S}_{i,j}(u,v)$ is interrupted.

If a particular depth of subdivision is obtained, the subdivision of the Bézier patch can be stopped, and the PP or LP solver for nonlinear polynomial systems in two variables can be applied to the Bézier patch. The equations for $\vec{S}_{i,j}(u,v)$ read as follows analogously to (2):

$$\sum_{k=0}^{2p-1}\sum_{l=0}^{2q} a_{k,l}B_{k,2p-1}(u)B_{l,2q}(v) = 0 \text{ and } \sum_{k=0}^{2p}\sum_{l=0}^{2q-1} b_{k,l}B_{k,2p}(u)B_{l,2q-1}(v) = 0$$

with

$$a_{k,l} = \sum_{s=\max(0,k-p)}^{\min(p-1,k)} \sum_{t=\max(0,l-q)}^{\min(q,l)} \frac{\binom{p-1}{s}\binom{p}{k-s}\binom{q}{t}\binom{q}{l-t}}{\binom{2p-1}{k}\binom{2q}{l}}(\vec{P}^{i,j}_{s+1,t} - \vec{P}^{i,j}_{s,t})(\vec{P}^{i,j}_{k-s,l-t} - \vec{Q}),$$

$$b_{k,l} = \sum_{s=\max(0,k-p)}^{\min(p,k)} \sum_{t=\max(0,l-q)}^{\min(q-1,l)} \frac{\binom{p}{s}\binom{p}{k-s}\binom{q-1}{t}\binom{q}{l-t}}{\binom{2p}{k}\binom{2q-1}{l}}(\vec{P}^{i,j}_{s,t+1} - \vec{P}^{i,j}_{s,t})(\vec{P}^{i,j}_{k-s,l-t} - \vec{Q}).$$

The combination of a classical formulation of the distance problem and the subdivision technique is recommended if a high subdivision depth is expected, e.g., if r in Theorem 2 is too large in case of a very bent Bézier curve.

§5. Concluding Remarks

The method described in this paper computes the distance between a point and a NURBS curve or surface. Our goal was to provide a reliable method to solve this problem. The first few subdivision steps and tests quickly locate the regions of potential solutions. Then the subdivision can be either continued, or one of the equation solvers or even a distance-calculation algorithm for polyhedra can be applied [4,5]. We have developed an interval version of the PP/LP algorithm using interval arithmetic and considering a correct handling of roots of order two, suitable modifications of Graham's scan algorithm for building the convex hull, the revised simplex method by Gass, and an adapted interval-based subdivision by de Casteljau. The solver has been implemented in C++ using the library Profil/BIAS (see [9]). This improves the robustness of the distance-algorithm, assures an interval enclosure of the solution, and makes it suitable for verification of off-line tasks in path planning.

Some modifications to the algorithms could improve performance, e.g., if upper bounds on the derivatives of order two for the curve or surface are known [6]. But doubtless our NURBS-based algorithm will be slower, e.g., compared with our algorithms [4,5], where we deal with the objects as polyhedra. In a more complete distance tracking system, such as a manipulator in a complex environment, a progressive switch from a spherical or polyhedral enclosure of the objects to NURBS surfaces is recommended, especially if contact problems are investigated.

Acknowledgments. This research was supported by Deutsche Forschungsgemeinschaft within the scope of Sonderforschungsbereich 291 "Elastic manipulation systems for heavy loads in complex environments".

References

1. Bobrow, J. E., A direct minimization approach for obtaining the distance between convex polyhedra, Int. J. Robotics Research **3**, Vol. 8 (1989), 65–76.
2. Cameron, S. and C. Turnbull, Computing distances between NURBS-defined convex objects, IEEE Int. Conf. Robotics & Autom. (1998), Leuven, 3685-3691.
3. Canny, J. and M. Lin, A fast algorithm for incremential distance calculation, IEEE Int. Conf. Robotics & Autom. (1991), Sacramento, 1008–1014.
4. Dyllong, E., W. Luther, and W. Otten, An accurate distance-calculation algorithm for convex polyhedra, Reliable Computing **3**, Vol. 5 (1999), 241-254.
5. Dyllong, E. and W. Luther, An accurate computation of the distance between a point and a polyhedron, to appear in special issues of ZAMM (Proceedings of GAMM'99, April 12-16, Metz).
6. Filip, D., R. Magedson, and R. Markot, Surface algorithms using bounds on derivatives, CAGD **3** (1986), 295–311.
7. Gilbert, E. G. and C-K. Foo, Computing the distance between general convex objects in three-dimensional space, IEEE Trans. Robotics & Automation **1**, Vol. 6 (1990), 53–61.
8. Gilbert, E. G., D. W. Johnson, and S. Keerthi, A fast procedure for computing the distance between complex objects in three-dimensional space, IEEE J. of Robotics & Automation **2**, Vol. 4 (1988), 193–203.
9. Küppel, O., BIAS - Basic interval arithmetic subroutines, Technical report 93.3, TU Hamburg-Harburg, July 1993.
10. Patrikalakis, N. M. and E. C. Sherbrooke, Computation of the solutions of nonlinear polynomial systems, CAGD **10** (1993), 379–405.
11. Patrikalakis, N. M., E. C. Sherbrooke, and J. Zhou, Computation of stationary points of distance functions, Engineering with Computers **9** (1993), 231–246.
12. Piegl, L. and W. Tiller, *The NURBS book*, Springer, 1995.
13. Sato, Y. et al., Algorithms for convex and non-convex objects, in Proc. IEEE Int. Conf. Robotics & Automation (1996), Minneapolis, 771–778.
14. Wang, G., The subdivision method for finding the intersection between two Bézier curves or surfaces, Zhejiang University J., Special Issue on Computational Geometry 1984 (in Chinese).
15. Wang, G. and W. Xu, The termination criterion for subdivision of the rational Bézier curves, Graphical Models and Image Processing **1**, Vol. 52 (1991), 93-96.

Eva Dyllong and Wolfram Luther, Dept. of Computer Science,
University of Duisburg, D-47048 Duisburg, Germany
dyllong,luther@informatik.uni-duisburg.de

Curves from Motion, Motion from Curves

Rida T. Farouki

Abstract. Geometry and kinematics have been intimately connected in their historical evolution and, although it is currently less fashionable, the further development of such connections is crucial to many computer–aided design and manufacturing applications. In this survey, we explore a variety of classical and modern problems that illustrate how simple rules of motion produce interesting curves and, conversely, the computational problems of generating motions with prescribed paths and speeds. These encompass the geometry of trajectories under centripetal forces; the transformation of rotary motion into motion along general curves by mechanisms; real–time curve interpolators for digital motion control; and the description of spatial motions that involve variations of both position and orientation. Such case studies illustrate some of the intellectual appeal, and practical importance, of a sustained dialog between the study of curves and of motions.

§1. Preamble

Our intent in this paper is to survey the intricate web of historical connections between geometry and kinematics, a theme that has played a key role in the development of mechanics and analysis. In contemplating this theme, we are obviously confronted by a profusion of interesting and fruitful topics — and we are thus obliged to adopt a rather anecdotal approach.

Apart from its intrinsic interest, we choose this subject with the hope of promoting greater synergy between modern–day problems of geometric design and motion control. Modern CAD systems are mainly concerned with creating "static" geometrical descriptions of artifacts, but the processes by which these artifacts are actually fabricated often involve complicated motions of a tool — e.g., a cutter in a milling machine, or a wire electrode in electrical discharge machining — relative to a workpiece. Compared to the sophistication of CAD models, current methods for motion planning in manufacturing processes are often rather crude and naive. Thus, there is much scope for securing greater precision and reliability, through the use of advanced mathematical methods, in the relatively undeveloped field of manufacturing geometry.

The symbiosis between geometry and kinematics has deep historical roots. Newton, in his *Quadrature of Curves* (1676), aptly characterizes it as follows:

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 63–90.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

> *Lines (curves) are described, and thereby generated, not by the apposition of parts but by the continued motion of points ... These geneses really take place in the nature of things, and are daily seen in the motion of bodies.*

However, this has not always been a happy union. Insofar as it embodies both *spatial* and *temporal* information, kinematics subsumes geometry. To upgrade a *curve* into a *motion* requires the ability to *rectify* (or measure arc lengths of) curves. As a basic philosophical tenet, Descartes held this to be impossible — see §3 below — and he sought to banish all curves whose definitions explicitly or implicitly assume rectifications from the "rigorous" domain of geometry to the nascent (and less–exact) science of mechanics. Although, in modern times, the philosophical/existential problem of rectification is no longer troublesome, we must still address the *computational* difficulties it entails (see §3).

The antithesis of Descartes' attempt to divorce geometry from kinematics would ultimately find its logical expression, in the context of the special theory of relativity, with Minkowski's introduction [42] of the concept of "space–time" as the most natural setting for the study of physical phenomena:

> *Henceforth space by itself, and time by itself, are doomed to fade away into mere shadows, and only a kind of union of the two will preserve an independent reality.*

The recent introduction of the Minkowski metric of space–time into problems of geometric design reveals a remarkable confluence of ideas concerning medial axis transforms, Pythagorean hodographs, envelopes, and offset curves [43].

In this survey we shall attempt, through a series of anecdotal sketches, to promote greater interest in the relationship between geometry and kinematics, and its application to CAD/CAM problems. We commence in §2 and §3 with a brief review of the manner in which curves may be defined, and the problems that measurement of arc length incurs. Perhaps the simplest motion is that of a particle experiencing a force toward a fixed center, of magnitude proportional to a power of the radial distance r. As is well–known, Newton showed that an r^{-2} force of gravity explains the conic form of planetary and cometary orbits. We shall see in §4, however, that this is just one aspect of a more profound theory of *motion under centripetal forces* in Newton's *Principia*.

Mechanisms such as gears, cams, and linkages are used to transform forces and motions in machinery. In §5 we discuss the four–bar linkage, a mechanism that directly transforms rotary motion into motion on a general curved path. CNC machines offer a more flexible approach to motion generation, based on sophisticated servo–systems that drive linear or rotary axes in a coordinated manner. In §6 we discuss the problem of *real–time interpolators*, which must accurately and efficiently interpret the path and speed information to generate "reference point" data required by the digital control algorithm.

The preceding examples are concerned with motion in Euclidean spaces. A motion that involves not just positional but also *orientational* coordinates (such as in 5–axis machining) may be regarded as the motion of a point in a higher–dimensional, non–Euclidean (soma) space. Some subtle problems that

arise with such motions are discussed in §7. Finally, §8 offers some concluding thoughts on our theme of connections between geometry and kinematics.

§2. Curves and Motions

Analytic geometry has its origins in the computational investigation of curves specified by suitable coordinate equations. There are basically two ways to define a plane curve in terms of Cartesian coordinates (x, y). We may select a *predicate function*, that indicates whether or not each point in the plane lies on the curve — this is typically a bivariate polynomial f in the coordinates, and the curve is the locus of points on which the polynomial vanishes:

$$f(x, y) = 0\,. \tag{1}$$

On the other hand, we may choose a pair of *generating functions*

$$x(t)\,,\; y(t) \tag{2}$$

that produce an ordered sequence of curve points when evaluated at successive values of a continuous "auxiliary variable" or parameter t.

Whereas the *implicit* description (1) is essentially "static," the *parametric* form (2) offers a more "dynamic" characterization of curves — it embodies the suggestion of motion along a curve, incurred by steady increase of the curve parameter t. It is a mistake, however, to invest too much hope in the capacity of parametric curves to adequately describe motions. Motion specification is concerned as much with the instants in time at which a body assumes given positions along a path, and corresponding *velocities* and *accelerations*, as with the path *geometry*. A motion is really a geometrical locus in Minkowski space, with one temporal and one or more spatial dimensions.

Of course, we can always *interpret* the parameter t as *time*, and equations (2) then completely specify *a* motion. However, if we wish to use only "simple" (polynomial or rational) functions, such motions are mathematical curiosities: except in trivial cases, they are neither solutions to appropriate equations of motion, nor do they represent motions of practical interest that we may wish to impose on a given locus. To emphasize that the curve parameter generally lacks any geometrical or temporal significance, we henceforth use the Greek character ξ to denote it, and we explicitly reserve t for *time*.

Connections between the study of curves and motions is a recurrent theme in the history of science and technology. At the inception of analytic geometry, motions offered an intuitive means to construct and analyze loci of increasing sophistication: see, for example, the remarkably diverse historical applications of the roulettes generated by the rolling motions of lines and circles (cycloids, circle involutes, epicycloids and hypocycloids, epitrochoids and hypotrochoids) discussed in [21]. Conversely, the conics of the ancient Greeks make a rather surprising appearance in the solution to the premier problem of dynamics: the determination of planetary orbits. In the modern computer era, the problem of producing a desired motion along a given path is central to real–time control of manufacturing, inspection, robotic, and other devices.

§3. Towards an Impossible Ideal

Since speed on a curved path is the rate of change of distance with time, the problem of rectification — i.e., the measurement of arc length — is evidently critical to the description of motion. This problem, however, has been fraught with computational difficulty since Descartes founded analytic geometry in an appendix *La géométrie* [15] to the *Discours de la méthode pour bien conduire sa raison et chercher la vérité dans les sciences* (1637). He asserts that:

> *Geometry should not include lines (curves) that are like strings, in that they are sometimes straight and sometimes curved, since the ratios between straight and curved lines are not known, and I believe cannot be discovered by human minds, and therefore no conclusion based upon such ratios can be accepted as rigorous and exact.*

Nevertheless, Descartes' dictum began to crumble almost immediately after its enunciation, amid a flurry of counter–examples.

For example, Galileo [26] realized that, when a body is dropped into a hole drilled through the center of a static Earth, it executes linear simple harmonic motion across the full Earth diameter under the influence of gravity:

> *... if the terrestrial globe were perforated through the center, a cannon ball descending through the hole would have acquired at the center such an impetus from its speed that it would pass beyond the center and be driven upward through as much space as it had fallen, its velocity beyond the center always diminishing with losses equal to the increments acquired in the descent ...*

On a rotating Earth, however, the body will have an initial tangential velocity, and the nature of its motion in the hypothetical case of "permeable" matter (which exerts gravitational forces but does not impede motion) is not obvious. Galileo's pupil, Evangelista Torricelli (1608–1647), conjectured that the path would be a logarithmic spiral about the Earth's center, described by

$$r = a\,\mathrm{e}^{k\theta} \tag{3}$$

in polar coordinates (also known as an "equi–angular" spiral, since the tangent makes a fixed angle, $\cot^{-1} k$, with the radius vector). Isaac Newton re–iterated this conjecture [2,60] in a letter dated November 28, 1679 to Robert Hooke, who criticized it during a Royal Society meeting the following December 11. As we shall see in §4 below, Torricelli and Newton were quite wrong: the path is actually — as intuitively argued by Hooke — an ellipse.

During his investigations, however, Torricelli discovered a rectification of the spiral (3) in 1645 through the Archimedean "method of exhaustion" — he showed that, for $-\infty < \theta \leq 0$, the arc length equals the length of the tangent at $\theta = 0$ extended to the y–axis [9] — namely, $\sqrt{1+k^{-2}}\,a$ (see Figure 1). This is a truly remarkable result, since the curve must execute an infinite number of gyrations about the origin before terminating there!

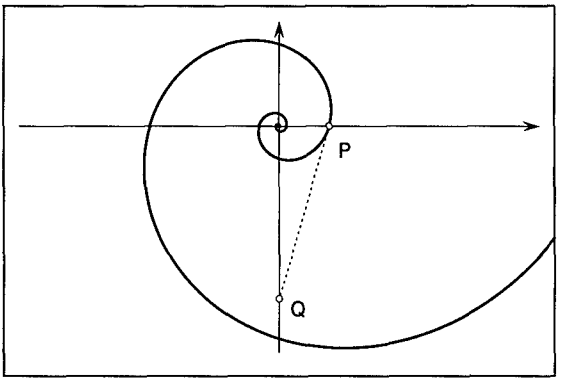


Fig. 1. The arc length of (3) for $\theta \leq 0$ equals the length PQ on the tangent line.

The logarithmic spiral was also known to Thomas Harriot (1560–1621) as the projection of a rhumb line on the Earth's surface — i.e., the path traced by a ship that sails in a fixed compass direction — onto the equatorial plane [55]. Jakob Bernoulli was so fascinated by the self-similarity of this curve under coordinated rotations and dilatations about the origin, that he arranged to have it engraved on his tombstone with the caption *Eadem mutata resurgo* [3] — "Though changed I shall arise the same."

Subsequently, another curve was rectified by Gilles Personne de Roberval (1602–1675) and Christopher Wren (1632–1723) — namely, the cycloid

$$x(\theta) = a(\theta - \sin\theta), \quad y(\theta) = a(1 - \cos\theta) \tag{4}$$

traced by a fixed point on a circle of radius a that rolls without slipping on a straight line (see Figure 2). They showed that a single "arch" $(0 \leq \theta < 2\pi)$ of this curve has length $8a$. Although it has now fallen into obscurity, the cycloid was a virtual "proving ground" for novel mathematical ideas and methods in the mid–17th century: it caught the attention of all the leading scientists, and prompted international competitions and acrimonious controversies. See [21] for a discussion of its tautochrone and brachistochrone properties.

To Descartes, however, the rectification of curves such as the spiral (3) and the cycloid (4) was suspect — they are not true "geometrical" (i.e., *algebraic*) but rather "mechanical" (i.e., *transcendental*) curves. By introducing angular variables, their definitions essentially *presuppose* a rectification (of the circle). Nonetheless, it was not long before even an algebraic curve, under the scrutiny of William Neil (1637–1670), Hendrick van Heuraet (1633–1660), and Pierre de Fermat (1601–1665), succumbed to rectification — the cuspidal cubic

$$x(\xi) = \xi^2, \quad y(\xi) = k\,\xi^3 \tag{5}$$

known as the "semicubical parabola." Its arc length s, measured from $\xi = 0$, is an *algebraic* function of the parameter:

$$s(\xi) = \frac{(9k^2\xi^2 + 4)^{3/2} - 8}{27k^2}.$$

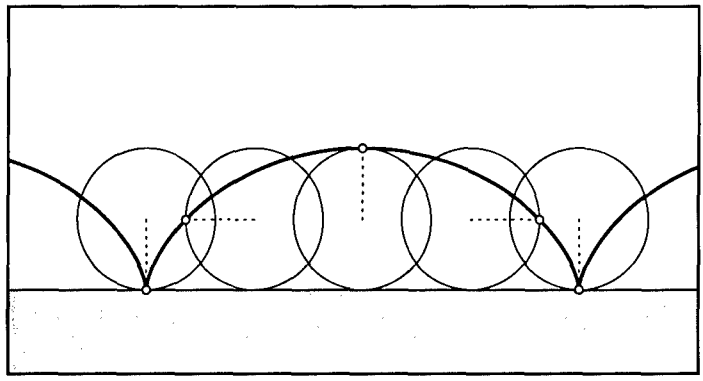


Fig. 2. Cycloid: the length of an arch is eight times the radius of the rolling circle.

Ironically, van Heuraet — an associate of Huygens — published his results in an appendix to van Schooten's 1659 Latin version of Descartes, *Geometria a Renato Des Cartes.* Neil's results also appeared in 1659, in the *Tractatus duo, prior de cycloide, posterior de cissoide* published by John Wallis, and Fermat's work followed in 1660 in *De linearum curvarum cum lineis rectis comparatione dissertatio geometrica* — an appendix to a treatise by de Lalouvère (this was the only publication by Fermat to appear during his lifetime).

Christiaan Huygens (1629–1695), in his *Horologium oscillatorium* of 1673, gave a historical account [34] of these rectifications that provoked arguments over the priority he attributed to van Heuraet and Wren for their discovery — see Chapter 8 of [31]. This dispute reflects the philosophical importance of the rectification problem, which had been held impossible through long tradition that originated with Aristotle, was reinforced in the 11th century by Ibn Rushd (Averroes), and culminated in Descartes' dogmatic assertion. Huygens' theory of evolutes and involutes, employed in his design of an isochronous pendulum clock [21], offered profound new insight into this age–old problem. The cubic (5) was recognized as the evolute (locus of centers of curvature) of a parabola, while the cycloid (4) has an identical (displaced) cycloid as its evolute.

All these results preceded a formal development of the calculus. Whereas the latter resolved existential issues concerning arc lengths by defining them, for a (sufficiently smooth) parametric curve $(x(\xi), y(\xi))$, through the integral

$$s(\xi) = \int_0^\xi \sqrt{x'^2(\tau) + y'^2(\tau)}\, \mathrm{d}\tau\,, \tag{6}$$

there remained the awkward fact that this does not admit analytic reduction except in trivial or exceptional cases, such as the cubic (5).

In fact, with the emergence of differential geometry, it became customary to assume $s \equiv \xi$ — i.e., the integrand in (6) is precisely unity — although this *natural* or *arc–length* parameterization has only a hypothetical existence: it is fundamentally incompatible with curves (except straight lines) parameterized by "simple" functions. This fact is obvious for polynomial curves, but its proof

for *rational* curves is subtle [23], involving Pythagorean triples of polynomials, partial fraction decompositions, and the calculus of residues.

An offshoot to this proof was the introduction of Pythagorean hodograph (PH) *curves*, whose hodograph components satisfy the condition

$$x'^2(\xi) + y'^2(\xi) \equiv \sigma^2(\xi)$$

for some polynomial $\sigma(\xi)$, and are thus [22] of the form

$$x'(\xi) = u^2(\xi) - v^2(\xi), \quad y'(\xi) = 2\,u(\xi)v(\xi), \quad \sigma(\xi) = u^2(\xi) + v^2(\xi)$$

where $u(\xi)$, $v(\xi)$ are relatively prime polynomials. For PH curves, the integral (6) evidently reduces to a *polynomial* function of the parameter ξ. This fact proves especially propitious in the formulation of *real–time CNC interpolators* for digital motion control applications (see §6 below).

Venturing beyond PH curves, one may seek to encompass a broader class of loci by allowing more complicated arc–length functions. Suppose we allow s to be an *algebraic function* of the parameter ξ — i.e., there exists a bivariate polynomial $F(\cdot,\cdot)$ such that (6) satisfies

$$F(s(\xi),\xi) = 0\,.$$

An algebraic function cannot, in general, be described by a simple closed–form expression. Nevertheless, one can show [53] that (6) is algebraic if and only if there exists a polynomial $h(\xi)$ such that

$$[\,x'^2(\xi) + y'^2(\xi)\,]\,h(\xi) \equiv h'^2(\xi)\,.$$

As an immediate consequence, if the function (6) is algebraic, it must have the simple form $s(\xi) = 2\sqrt{h(\xi)} + \text{constant}$ (note that the PH curves are subsumed as special instances, corresponding to $h = \frac{1}{4}[\int u^2 + v^2\,\mathrm{d}\xi\,]^2$). The cubic (5) is the simplest (non–PH) example of these *algebraically–rectifiable curves*, with $h(\xi) = (9k^2\xi^2+4)^3/2916k^4$ — indeed, it is the *unique* cubic with this property. See [53] for details on algebraically–rectifiable quartics and quintics.

Since arc–length parameterization by rational functions is fundamentally impossible, it seems natural to ask "how close" we can approach this elusive ideal. Consider, for example, a degree–n polynomial curve $\boldsymbol{r}(\xi)$: a parameter transformation $\xi \in [\,0,1\,] \rightarrow \tau \in [\,0,1\,]$ of the form

$$\xi = \frac{(1-\alpha)\tau}{\alpha(1-\tau) + (1-\alpha)\tau}$$

gives a rational representation of the same degree, and offers a single degree of freedom, α, to control the "parameter flow" over the curve. Using the integral

$$I = \int_0^1 (|\boldsymbol{r}'(\tau)| - 1)^2\,\mathrm{d}\tau \tag{7}$$

as a measure of "closeness" to arc–length parameterization (for which $I = 0$), the value of α that minimizes (7) can be found [18] as the unique root on $(0, 1)$ of a quadratic equation; see also [36]. However, this optimal parameterization offers limited scope for improvement, since we fix the curve degree n.

Another approach, based on the polynomial arc–length functions $s(\xi)$ of PH curves, employs the Legendre series to compute a convergent sequence of (constrained) polynomial approximations $\xi_1(s), \xi_2(s), \ldots$ to the *inverse* of this function [19], such that

$$\lim_{k\to\infty} \xi_k(s(\xi)) \equiv 1 \qquad \text{for } \xi \in [0,1]\,,$$

given the normalization $s \in [0,1]$. The coefficients of $\xi_k(s)$ can be determined through closed–form reduction of certain integrals. For sufficiently high k, the re–parameterized version $\boldsymbol{r}_k(s) = \boldsymbol{r}(\xi_k(s))$ comes arbitrarily close to the exact arc–length parameterization, although it is formally of degree kn.

§4. Curves from Motion I. Centripetal Forces

By integrating the description of the forces and laws of motion that govern a physical system, the science of dynamics provides a rigorous and quantitative approach to analyzing motions. Perhaps more so than in any other branch of science [12], the theoretical canonization and empirical triumph of dynamics are the fruits of a single pre–eminent mind: Sir Isaac Newton.

Perhaps the simplest (non–trivial) problem of dynamics is that of motion under a centripetal force — i.e., a force always directed toward or away from a fixed center, whose magnitude depends only on distance r from that center. The term *centripetal* — "seeking the center" — was introduced [60] by Newton in his *De motu corporum in gyrum* of 1684 (in recognition of the fact that, to overcome the inertial tendency of a body to move in a straight line, circular motion requires a steady force directed toward a fixed center):

> *I call that, by which a body is impelled or attracted toward some point which is regarded as a center, centripetal force.*

The basic questions concerning centripetal forces are: *what kinds of orbit arise from different dependencies of the force on* r — and, conversely, *knowing the type of orbit, can we deduce the dependence of the force on* r?

In the late 17th century the context for interest in such questions was, of course, the search for an explanation of Kepler's (empirical) laws of planetary motion — namely: (i) the orbits of the planets are ellipses, with the sun at one focus; (ii) the radial line between the sun and a planet sweeps out area at a uniform rate; and (iii) the squares of the orbital periods are proportional to the cubes of the mean distances of planets from the sun. As we now know, these are direct consequences of an *inverse–square* (r^{-2}) gravitational force. Newton discovered this at an early stage in his career, but remained characteristically secretive about it. It is Edmond Halley who deserves credit for coaxing Newton into disseminating his arguments and, ultimately, codifying dynamics through publication of the *Principia*. According [13] to Abraham de Moivre:

> *In 1684 D^r Halley came to visit him at Cambridge, after they had been some time together the D^r asked him what he thought the Curve would be that would be described by the Planets supposing the force of attraction towards the Sun to be reciprocal to the square of their distance from it. S^r Isaac replied immediately it would be an Ellipsis, the D^r struck with joy & amazement asked him how he knew it, why saith he, I have calculated it, whereupon D^r Halley asked him for his calculation without any further delay, S^r Isaac looked among his papers but could not find it, but he promised to renew it, & then send it him ...*

In fact Halley, as Clerk to the Royal Society, printed the *Principia* at his own expense; the Society's funds had been depleted by the production of a *Historia Piscium* (*History of Fishes*) that failed to become a best–seller. Subsequently, Halley's salary was paid entirely in copies of this *Historia Piscium* [13].

The inverse–square nature of gravitational force, now common knowledge, was established by Newton as the rational explanation for Kepler's laws. It is not widely known, however, that the *Principia* thoroughly analyzes a variety of power–law (r^n) centripetal forces, and shows that different integer exponents n yield circular, conic, spiral, and other orbits in an often surprising manner. The profundity of these results — and their anticipation of an elegant theory of *dual centripetal forces* due to K. Bohlin [7] and E. Kasner [38] — has recently been emphasized by Arnol'd [2], Chandrasekhar [10], and Needham [45].

As reflected in the title of Chandrasekhar's recent book [10], the obscurity of such interesting results in the *Principia* is due to Newton's exclusive reliance on forbiddingly Euclidean argumentations. The modern reader, equipped with predominantly analytic/algebraic skills, is usually reduced to a state of dismay and bewilderment upon a first encounter with the *Principia*. Needham [45] argues convincingly that the re–discovery of Newton's "geometrical calculus" is a very rewarding endeavor, and Arnol'd [2] gives an example of the type of problem — the transcendental nature of certain area integrals — that seems "obvious" to Newtonian thinking, but not to modern modes of thought.

To elucidate connections between the *Principia*'s results and the Bohlin–Kasner theory of dual centripetal forces, it is convenient [2,44] to adopt the complex–number representation $z = x + \mathrm{i}\,y = r\,\mathrm{e}^{\mathrm{i}\theta}$ of the Euclidean plane. Under a centripetal force proportional to the n–th power of distance ¿from the origin, the equation of motion is then

$$\frac{\mathrm{d}^2 z}{\mathrm{d}t^2} = -\,k\,|z|^n\,\frac{z}{|z|}\,. \tag{8}$$

Here, the centripetal force is attractive or repulsive according to whether k is positive or negative. Hooke's law, for example, corresponds to $n = +1$, while $n = -2$ represents a Coulomb (gravitational or electrostatic) force.

There are two "constants of motion" associated with solutions $z(t)$ to the differential equation (8) — the angular momentum and energy,

$$L = \mathrm{Im}\left(\overline{z}\,\frac{\mathrm{d}z}{\mathrm{d}t}\right) = r^2\,\frac{\mathrm{d}\theta}{\mathrm{d}t} \qquad \text{and} \qquad E = \frac{1}{2}\left|\frac{\mathrm{d}z}{\mathrm{d}t}\right|^2 + k\,\frac{|z|^{n+1}}{n+1}\,. \tag{9}$$

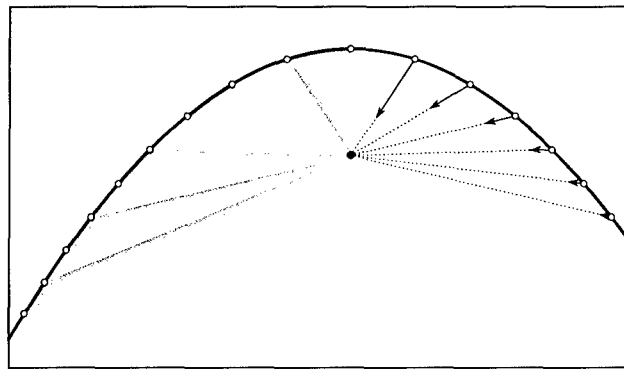


Fig. 3. Constancy of angular momentum L under an r^{-2} centripetal force.

Although we may formally replace $|z|^{n+1}/(n+1)$ by $\ln|z|$ when $n = -1$, this case is usually excluded. For $n \leq -2$, the "potential energy" component of E is naturally negative, and tends to zero as $|z| \to \infty$. Conversely, for $n \geq 0$, the potential energy is naturally positive and vanishes when $|z| = 0$. The function $\ln|z|$, however, diverges for both $|z| \to 0$ and $|z| \to \infty$; it does not represent a satisfactory scale–free potential energy with a natural reference value.

The polar form of L given in (9) serves as a reminder of its geometrical interpretation, namely, the rate at which the position vector sweeps out area in the orbit — this constancy of "areal velocity" is expressed by Kepler's second law. Section II of Book I of the *Principia* is devoted to "the determination of centripetal forces," and Newton's immediate concern [46] is to prove:

PROPOSITION I. THEOREM I

The areas which revolving bodies describe by radii drawn to an immovable centre of force do lie in the same immovable plane, and are proportional to the times in which they are described.

PROPOSITION II. THEOREM II

Every body that moves in any curved line described in a plane, and by a radius drawn to a point either immovable, or moving forwards with an uniform rectilinear motion, describes about that point areas proportional to the times, is urged by a centripetal force directed to that point.

In other words, Newton first shows that "centripetal force $\Longleftrightarrow$ L = constant" (this principle is illustrated in Figure 3, for the case of a Coulomb r^{-2} force).

Consider now the conformal map $z \to w$ of the plane given by

$$w = z^{\alpha}, \tag{10}$$

under which the orbit $z(t)$ determined by equation (8) is transformed into an orbit $w(\tau)$, where t and τ denote times on these orbits corresponding to an

angular position θ about the origins of the z and w planes. Writing $z = r\,e^{i\theta}$ and $w = \rho\,e^{i\theta}$, we stipulate that these orbits have equal angular momentum

$$L = r^2 \frac{d\theta}{dt} = \rho^2 \frac{d\theta}{d\tau} = \text{constant}, \tag{11}$$

and we ask: *is the orbit $w(\tau)$ also the solution to an equation of motion*

$$\frac{d^2 w}{d\tau^2} = -\kappa |w|^m \frac{w}{|w|} \tag{12}$$

under a power-law centripetal force — and if so, how are the exponents n, m, α related? It transpires that this problem has an elegant and *unique* solution: corresponding force–law exponents n and m are related by

$$(n+3)(m+3) = 4, \tag{13}$$

and the exponent α of the map (10) is given in terms of them by

$$\alpha = \frac{n+3}{2} = \frac{2}{m+3}. \tag{14}$$

To derive equations (12)–(14), we first note from (10) and (11) that derivatives with respect to t and τ are related by

$$\frac{d}{d\tau} = |z|^{2(1-\alpha)} \frac{d}{dt}. \tag{15}$$

Now by applying (15) to $w = z^\alpha$ twice, and invoking (8), we obtain

$$\frac{d^2 w}{d\tau^2} = 2\alpha(1-\alpha)\left[\frac{1}{2}\left|\frac{dz}{dt}\right|^2 + k\frac{|z|^{n+1}}{2(\alpha-1)}\right]|z|^{2-3\alpha}\frac{z^\alpha}{|z^\alpha|}.$$

We observe that by choosing $2(\alpha-1) = n+1$, i.e., $\alpha = (n+3)/2$ as in (14), the expression in brackets coincides with the energy constant E in (9). With this choice, substitution from (10) gives

$$\frac{d^2 w}{d\tau^2} = -2\alpha(\alpha-1)E\,|w|^{(2-3\alpha)/\alpha}\frac{w}{|w|}.$$

This is of the desired form (12), with $\kappa = 2\alpha(\alpha-1)E$ and $m = (2-3\alpha)/\alpha$, i.e., $\alpha = 2/(m+3)$ as in (14). Finally, equation (13) follows from the individual relations in (14), between n and α, and m and α, derived above.

Equation (13) describes a hyperbolic relation between the force exponents n and m. For each n, except -3, the orbit $z(t)$ determined by (8) is mapped by (10) into a dual orbit $w(\tau)$ of equal angular momentum, determined by (12).

Dual orbits corresponding to *integer* force exponents are of special interest — they are (with $n \geq m$) as follows:

$$\begin{aligned} &\text{(a)} \quad n = +1, \quad m = -2, \quad z \to z^2, \\ &\text{(b)} \quad n = -1, \quad m = -1, \quad z \to z, \\ &\text{(c)} \quad n = -4, \quad m = -7, \quad z \to z^{-1/2}, \\ &\text{(d)} \quad n = -5, \quad m = -5, \quad z \to z^{-1}. \end{aligned}$$

Cases (b) and (d) identify "self–dual" forces (as noted above, however, case (b) is usually excluded on physical grounds). Case (a) reveals the beautiful result that an orbit $\boldsymbol{z}(t)$ under a linear Hooke's–law force is mapped by $\boldsymbol{w} = \boldsymbol{z}^2$ to an orbit $\boldsymbol{w}(\tau)$ under an inverse–square Coulomb force. Indeed, Newton shows that the Hooke and Coulomb forms are the *only* centripetal forces that admit conic orbits. In this regard, see [44] for an interesting anecdote concerning the £1 note issued to commemorate the *Principia*'s 300th anniversary.

To investigate the geometry of orbits under power–law centripetal forces, it is convenient to employ polar coordinates (r, θ). Using the fact that $r^2\dot{\theta} = L$, we can write the scalar components of the equation of motion (8) as

$$\ddot{r} - L^2 r^{-3} + k r^n = 0 \qquad \text{and} \qquad 2\dot{r}\dot{\theta} + r\ddot{\theta} = 0\,, \tag{16}$$

where dots denote time derivatives. To eliminate the time variable, and obtain a purely geometrical description of the orbit, we set $u = 1/r$ and note that

$$\frac{\mathrm{d}}{\mathrm{d}t} = \dot{\theta}\frac{\mathrm{d}}{\mathrm{d}\theta} = Lu^2\frac{\mathrm{d}}{\mathrm{d}\theta}\,.$$

With $\beta = k/L^2$, the first of equations (16) can then be transformed [10,57] to

$$u'' + u - \beta u^{-(n+2)} = 0\,, \tag{17}$$

where primes denote derivatives with respect to θ. By solving this differential equation, we obtain polar–coordinate expressions $r(\theta) = 1/u(\theta)$ describing the shapes of orbits. One may verify the functional form and geometrical nature of some representative solutions known to Newton:

$$\begin{aligned} &n = +1: \quad r(\theta) = (\beta \sin^2\theta + \cos^2\theta)^{-1/2} && \text{ellipse, center at origin;} \\ &n = -2: \quad r(\theta) = (\beta + \cos\theta)^{-1} && \text{ellipse, focus at origin;} \\ &n = -3: \quad r(\theta) = \exp\left(\sqrt{\beta - 1}\,\theta\right) && \text{logarithmic spiral;} \\ &n = -5: \quad r(\theta) = \sqrt{\beta/2}\,\cos\theta && \text{circle through origin.} \end{aligned}$$

The cases $n = 1$ and -3 are treated in PROPOSITION X. PROBLEM V and PROPOSITION IX. PROBLEM IV of the *Principia*, Book I. The case

$n = -5$ is pathological, since the orbit passes *through* the center of force! In PROPOSITION V. PROBLEM II, Newton actually treats a generalization of the $n = -5$ case — he shows that a particle $\boldsymbol{p}$ will execute a circular orbit if it is attracted to any center $\boldsymbol{c}$ by a force proportional to $r^{-2}\ell^{-3}$, where r is the distance of $\boldsymbol{p}$ from $\boldsymbol{c}$, and ℓ is the length of the chord containing $\boldsymbol{p}$ and $\boldsymbol{c}$. When $\boldsymbol{c}$ lies *on* the circle, we have $\ell = r$, and hence an r^{-5} force.

In Section III of Book I, Newton is concerned with "the motion of bodies in eccentric conic sections." He treats the case $n = -2$ in PROPOSITION XI. PROBLEM VI, and also discusses parabolic and hyperbolic orbits. Kepler's third law is also derived, in PROPOSITION XV. THEOREM VII.

Incidentally, the case $n = 1$ provides the correct solution to the problem of motion in the gravity of a permeable rotating Earth, considered by Torricelli (see §3). If the Earth is a homogeneous sphere of mass M and radius R, the gravitational force at distance r from the center is equal to

$$\frac{GM(r/R)^3}{r^2},$$

G being the gravitational constant — i.e., it is proportional to r. Thus, the path is an ellipse, and not the logarithmic spiral suggested by Torricelli (which requires an r^{-3} force). Actually, it is a very *shallow* ellipse — the minor axis is smaller than the major axis R by the dimensionless factor $\sqrt{GM/\omega^2 R^3} \approx 291$, where $\omega = 2\pi$ rads/day (this factor is the ratio of the orbital velocity at $r = R$ to the tangential velocity ωR at the equator due to the Earth's rotation).

A shallow ellipse is an obvious perturbation to Galileo's simple harmonic motion through a non–rotating permeable Earth, and in retrospect Torricelli's conjectured spiral trajectory — revived by Newton in 1679 — may seem rather naive. Newton soon redeemed himself, however, through the publication of his *Principia* in 1686, which contains the correct solution as part of a remarkably comprehensive theory of orbital motions under centripetal forces.

It is a sobering experience, for the modern reader, to pierce the *Principia*'s veil of geometrical argumentations, and appreciate its profound insights. To contemporaries, Newton's creation was a virtually miraculous event — Halley composed an ode to preface the *Principia*, extolling "the illustrious man" and his work, "a signal distinction of our time and race" [46]:

Matters that vexed the minds of ancient seers,
And for our learned doctors often led
To loud and vain contention, now are seen
In reason's light, the clouds of ignorance
Dispelled at last by science ...

However, Newton was not exempt from the sarcasm of critics, such as the poet Alexander Pope [51], who were often more eloquent in their converse views:

Superior beings, when of late they saw
A mortal man unfold all Nature's law,
Admired such wisdom in an Earthly shape,
And showed a NEWTON as we show an ape.

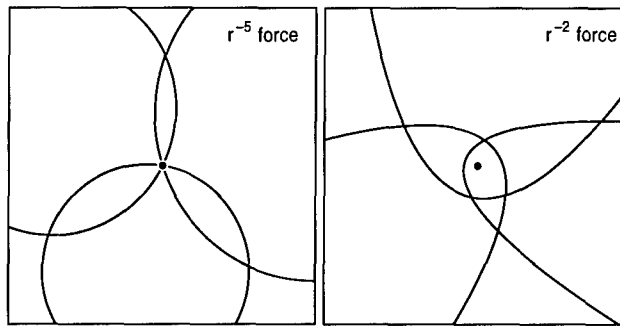


Fig. 4. Geometry of zero-energy orbits under r^{-5} and r^{-2} centripetal forces.

To conclude, we note that the singular orbit $r(\theta) = \sqrt{\beta/2}\,\cos\theta$, identified above for an r^{-5} force, corresponds to the case where the orbital energy

$$E = \frac{\dot{r}^2}{2} + \frac{L^2}{2r^2} - \frac{k}{4r^4}$$

is zero: it is the analog of zero-energy parabolic orbits under an r^{-2} force (see Figure 4). As $r \to 0$ ($t \to \pi k/8L^3$), the positive kinetic energy and negative potential energy both become infinite in a manner such as to maintain $E = 0$. The angular momentum L is also conserved, in a limiting sense, as $r \to 0$.

For orbits with $E \neq 0$, equation (17) can be integrated to obtain

$$\sqrt{2}\int_{1/r_0}^{1/r} \frac{\mathrm{d}u}{\sqrt{\beta u^4 - 2u^2 + \gamma}} = \theta$$

where $\gamma = E/L^2$ and $r = r_0$ for $\theta = 0$. A further reduction, giving r explicitly in terms of θ, is possible upon introducing Jacobian elliptic functions [41], but we shall not pursue it here. A special case is a circular orbit $u = u_0 = 1/r_0$, of energy $E = \frac{1}{4}(2L^2 - ku_0^2)u_0^2$, but this is highly unstable — any perturbation will cause r to rapidly decay to 0 or grow to ∞ (see Figure 5, comparing orbits under r^{-5} and r^{-2} forces perturbed by introducing an initial negative/positive radial velocity corresponding to a 10^{-3} fractional change in E).

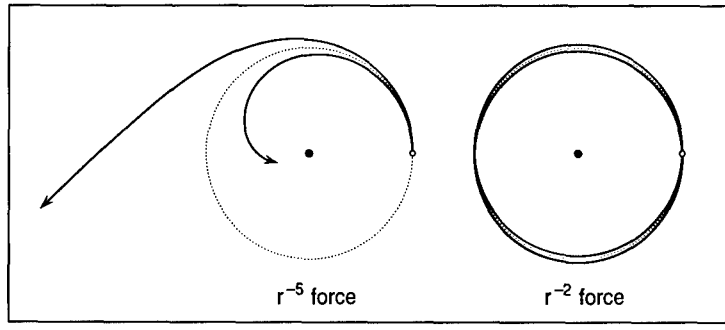


Fig. 5. Relative stability of circular orbits under r^{-5} and r^{-2} centripetal forces.

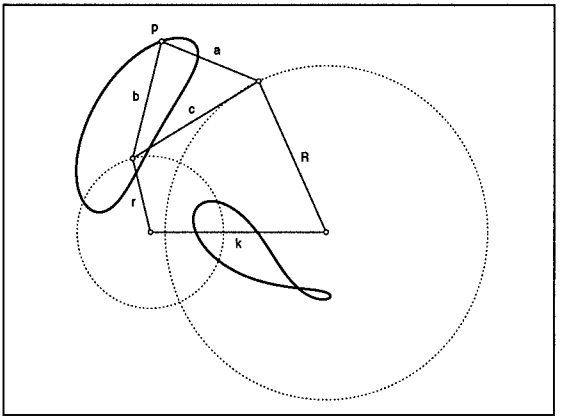


Fig. 6. Geometrical dimensions of a planar four–bar linkage.

In fact, only the $n = +1$ and -2 force laws admit *stable* periodic orbits, in the sense that perturbations to them always produce similar, "neighboring" closed orbits (a proof is given in Appendix A of Goldstein [28]).

§5. Curves from Motion II. Four–bar Linkage

A mechanism is a device that is designed to transform "input" motions and forces, from a given power source, into "output" motions and forces — better suited for use in some practical application. A mechanism typically comprises several rigid members connected by joints that allow certain types of relative motion. According to the Kempe theorem [39], mechanisms that employ only revolute and prismatic joints can (in principle) be designed to produce motion along *any* plane algebraic curve. We consider here the coupler curves of planar four–bar linkages, which serve to transform a rotational input motion into an output motion along some general curved trajectory.

(In early studies, such mechanisms were called *three*–bar linkages, since the "ground" link was not counted. It is now customary to include it, to give a closed kinematic chain. The idea of kinematic chains was introduced by the German engineer Franz Reuleaux, in his *Theoretische Kinematik* of 1875).

Figure 6 shows the geometrical configuration of a four–bar linkage. Such mechanisms are found in diverse contexts (windshield wipers, electric shavers, cranes, etc.). Historically, the most famous example was the "parallel motion" mechanism devised by James Watt (1736–1819) for his double–acting steam engine of 1782. In Figure 6, the link of length k is held fixed, while links of length r and R pivot about its two endpoints. These links are connected by a further link of length c, whose ends are thus constrained to lie on circles of radii r and R centered on $(0, 0)$ and $(k, 0)$. A point at a fixed position relative to the link of length c thus traces a locus, called the coupler curve, when the links of length r and R rotate (see Figure 7). We identify a specific point $\boldsymbol{p}$ by taking it to be the apex of a triangle of sides a and b, with base c.

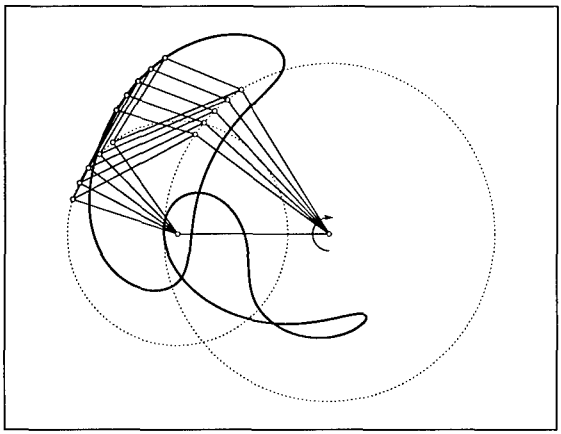


Fig. 7. Generation of a coupler curve by a four–bar linkage.

According to the Grashof theorem [29], the sum of lengths of the shortest and longest links should not exceed the sum of lengths of the other two links, if there is to be continuous relative rotation between two links. This condition is satisfied by the configuration shown in Figure 6, a *crank–rocker* mechanism: the link of length r (the crank) completes full revolutions, while that of length R (the rocker) oscillates through partial revolutions; the link of length c is the coupler. Other possible mechanisms — the *double–crank* or *double–rocker* — are obtained by varying the lengths k, r, R, and c of the links.

Four–bar linkages are capable of generating a rich variety of curved paths: the Hrones–Nelson "atlas" [32], for example, illustrates over *seven thousand* different forms of the coupler curve! These paths are all (parts of) an algebraic curve of degree 6, that depends on six parameters — the dimensions k, r, R, a, b, and c. Its equation can be succinctly expressed [4] in the form

$$f(x,y) \;=\; u^2(x,y) \,+\, v^2(x,y) \,-\, w^2(x,y) \;=\; 0 \tag{18}$$

where

$$\begin{aligned} u(x,y) \;&=\; a\,[\,(x-k)\cos\gamma + y\sin\gamma\,]\,(x^2+y^2+b^2-r^2) \\ &\quad -\, bx\,[\,(x-k)^2+y^2+a^2-R^2\,]\,, \\ v(x,y) \;&=\; a\,[\,(x-k)\sin\gamma - y\cos\gamma\,]\,(x^2+y^2+b^2-r^2) \\ &\quad +\, by\,[\,(x-k)^2+y^2+a^2-R^2\,]\,, \\ w(x,y) \;&=\; 2ab\,[\,x(x-k)\sin\gamma + y^2\sin\gamma - ky\cos\gamma\,]\,, \end{aligned} \tag{19}$$

and we have introduced the angle $\gamma = \cos^{-1}(a^2+b^2-c^2)/2ab$ in lieu of c. The curve defined by (18) and (19) has an ordinary triple point at each of the two circular points at infinity, and three affine double points (two of which may be complex conjugates) that always lie [4] on the circle

$$x^2 \,-\, kx \,+\, y^2 \,-\, ky\cot\gamma \;=\; 0\,, \tag{20}$$

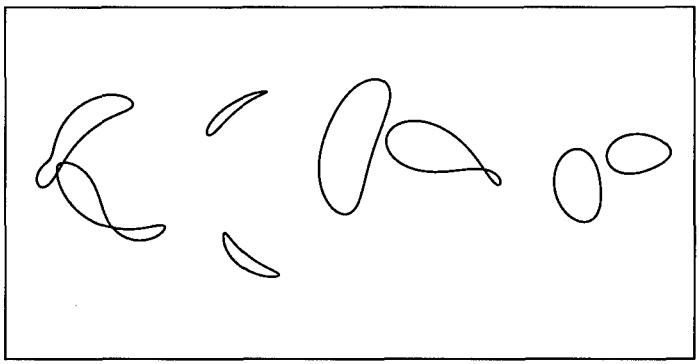


Fig. 8. Coupler curves for various values of the parameters k, r, R, a, b, c.

illustrated in the example in Figure 9 below. The coupler curve falls one short of the maximum of 10 double points that an algebraic curve of degree 6 may have, and is therefore of genus 1 — i.e., it is an *elliptic* curve.

For a crank–rocker, the curve defined by (18) and (19) always comprises two real loops. The physical mechanism traces just one of them: to trace the other loop, the initial configuration of the linkage must be changed. Further examples of crank–rocker coupler curves are shown in Figure 8. For double–crank and double–rocker mechanisms, the coupler curve also has two loops, but in the latter case the mechanism cannot trace either loop entirely. Four–bar linkages that do *not* satisfy the Grashof condition exhibit single–loop coupler curves, which may self–intersect (as in the Figure 9 example — note that the mechanism cannot trace the entire curve). Equations (18) and (19) encompass all these forms for suitable choices of k, r, R, a, b, and c.

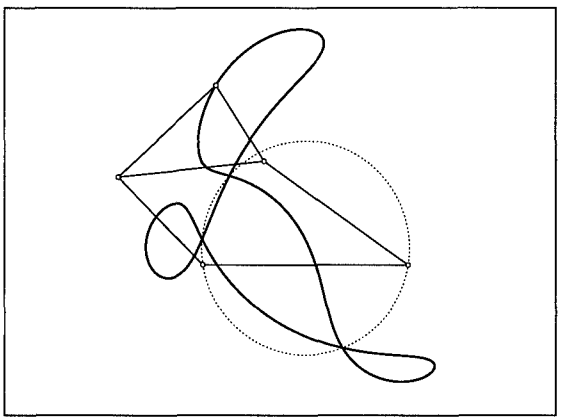


Fig. 9. Single–loop coupler curve for a non–Grashof mechanism.

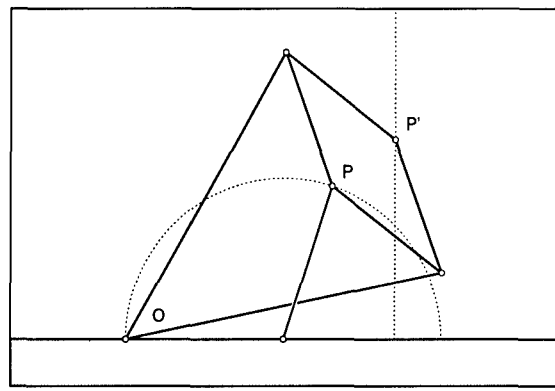


Fig. 10. Peaucellier mechanism: circular motion of P yields linear motion of P'.

A remarkable property of four–bar linkage coupler curves is expressed by the Roberts–Chebyshev theorem [50] — a given coupler curve may actually be traced by *three different four–bar mechanisms* (which are said to be cognates with respect to the given curve). This is not merely a mathematical curiosity: once a path is realized through a specific mechanism, one of its cognates may be found to produce better force transmission characteristics.

The design of a mechanism for a given path may be based on consultation of an "atlas" of coupler curves [32], or use of numerical methods to find the linkage dimensions that will give a locus interpolating prescribed points [6,58]. Allowing for freedoms in the choice of coordinate system, the general coupler curve defined by (18) and (19) can be made to interpolate nine points, but determining the mechanism parameters involves solving a formidable system of non–linear algebraic equations. Wampler et al. [58] have shown, for example, that (counting cognates) the nine–point problem has 4326 solutions, many of which may correspond to complex values for the link dimensions, or interpolate the discrete points on incompatible portions of the coupler curve.

As previously noted, one of the first applications of four–bar linkages was to transform the reciprocating linear motion of a piston into rotary motion of a shaft. Watt, Chebyshev, Roberts, and others proposed approximate solutions to this problem, but linkages that offer *exact* transformations between linear and circular motion were not known until 1864, when a captain in the French army, A. Peaucellier, devised the mechanism shown in Figure 10, comprising four links of length a and two of length b ($> a$). If the point P is constrained to move on a circle passing through the pivot O, the point P' traces a straight line, such that O, P, P' always remain collinear, and the distances r and r' of P and P' from O satisfy $rr' = b^2 - a^2$. Thus, P and P' are images of each other under *inversion* in the circle with center O and radius $\sqrt{b^2 - a^2}$.

Further details on the geometrical and kinematical properties of coupler curves may be found in standard texts [4,16,30,33] on kinematics — see also [47] for an interesting history of coupler curve synthesis methods.

§6. Motion from Curves I. Multi–axis CNC Machines

We have described above how interesting curves can arise from motions under specified kinematical or dynamical constraints. Computer numerical control (CNC) technology is concerned with the converse problem — i.e., the physical realization (by cutting tools, robot arms, sensors, etc.) of motions specified by geometrical paths and given speeds or feedrates along these paths.

To produce a desired motion, a CNC machine must drive each of its axes in an independent but coordinated manner. The controller algorithm employs digital representations of space and time: the unit of time, or sampling interval (typically $\sim$ 1 millisecond), is defined by a "clock" within the algorithm, while the *basic length unit* (BLU, typically $\sim$ 10 microns), or spatial resolution of the machine, is determined by position encoders mounted on its axes.

Within each sampling interval Δt, the controller must compare the actual position of each axis (as measured by the encoders) with the intended position (computed from the specified paths and feedrates by a real–time interpolator). The discrepancy between the actual and desired positions is used to generate control signals for the machine drives, ensuring that the specified paths/speeds are accurately realized. The discrete positions on a curved path $\boldsymbol{r}(\xi)$ computed by the interpolator are known as reference points — they are identified by the sequence $\xi_0, \xi_1, \xi_2, \ldots$ of parameter values satisfying $\xi_0 = 0$ and

$$\int_{\xi_{k-1}}^{\xi_k} \frac{\sigma \, d\xi}{V} = \Delta t \qquad \text{for } k = 1, 2, \ldots , \tag{21}$$

where $\sigma(\xi) = |\boldsymbol{r}'(\xi)|$ is the parametric speed of the curve and V is the (constant or variable) feedrate. Since the integral does not ordinarily have a closed–form reduction, and the unknowns are *limits of integration*, equation (21) is difficult to solve accurately and efficiently — even if V is constant.

Because of this computational difficulty, it is customary to employ simple (piecewise–linear/circular) "G code" approximations to curved tool paths [1]. Compared to its electromechanical hardware sophistication, the part program data that drives a CNC machine is embarrassingly crude. Some authors [11,61] have proposed to drive CNC machines along general curved paths by invoking approximate solutions to (21), based on the Taylor–series expansion

$$\xi_k = \xi_{k-1} + \frac{V}{\sigma}\Delta t + \frac{V}{\sigma^2}\left(V' - \frac{\boldsymbol{r}' \cdot \boldsymbol{r}''}{\sigma^2} V\right)\frac{(\Delta t)^2}{2} + \cdots , \tag{22}$$

where primes indicate derivatives with respect to the curve parameter ξ, and it is understood that σ, $\boldsymbol{r}'$, $\boldsymbol{r}''$, V, V', etc., are evaluated at ξ_{k-1}.

The extension of (22) to cubic and higher–order terms incurs complicated coefficients, and is thus ill–suited to real–time computation. Truncation errors are inevitable with this approach (most implementations, in fact, retain only the linear term). Note also that, for a non–constant feedrate, the variation of V cannot be usefully specified as a function of ξ. It must be given in terms of

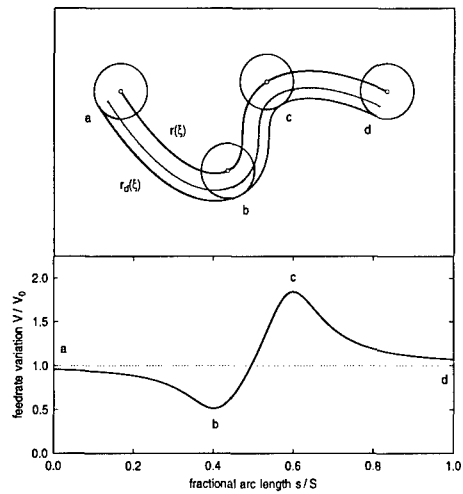


Fig. 11. Curvature dependent feedrate (24) for a constant material removal rate.

a physically meaningful variable, such as time t, arc length s, or curvature κ — in terms these variables, the derivative of V in (22) may be written as

$$V' = \frac{\sigma}{V}\frac{\mathrm{d}V}{\mathrm{d}t} = \sigma\frac{\mathrm{d}V}{\mathrm{d}s} = \sigma\frac{\mathrm{d}\kappa}{\mathrm{d}s}\frac{\mathrm{d}V}{\mathrm{d}\kappa}\,.$$

The Pythagorean–hodograph curves offer an elegant and rigorous solution to this dilemma [24]. For PH curves, the integral (21) admits a simple analytic reduction — not only for constant feedrate, but also varying feedrates specified in a number of useful ways, such as:

(a) any function $V(t)$ of *time* with known indefinite integral;
(b) a linear or quadratic polynomial $V(s)$ in the *arc length*;
(c) simple rational expressions $V(\kappa)$ in the local *curvature*;
(d) constant feedrate V along an *offset* to a specified curve.

In (almost) all these cases, the interpolation equation (21) reduces to the form

$$s(\xi_k) = F(\ldots, \xi_{k-1})\,, \tag{23}$$

where $s(\xi)$ is the polynomial arc–length function of the PH curve, and F is a known elementary function of the parameters describing the feedrate variation and the preceding reference point ξ_{k-1}. Since $s(\xi)$ is a *monotone* polynomial, equation (23) has a unique real root for the value of ξ_k, which may be obtained to machine precision by a few Newton–Raphson iterations starting from ξ_{k-1}.

The ability to perform real–time interpolation with continuously varying feedrates is extremely useful in a variety of practical problems. For example, the curvature–dependent feedrate

$$V(\kappa) = \frac{V_0}{1 + \kappa(d - \frac{1}{2}\delta)} \tag{24}$$

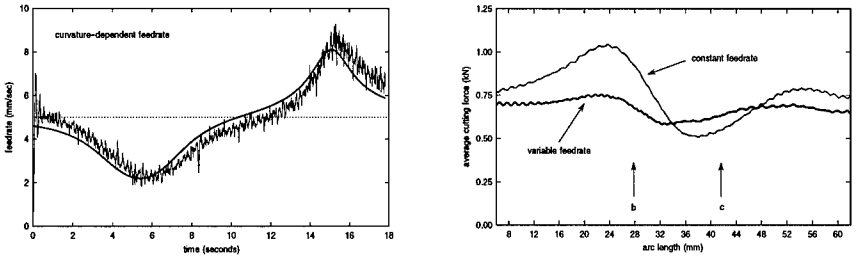


Fig. 12. Measured feedrate (left) and cutting force (right) using the function (24).

can suppress machining force variations [20] when a fixed depth of cut δ is to be removed along a curved path $\boldsymbol{r}(\xi)$ with a tool of radius d (see Figure 11: the material removal rate is higher at the "concave" location b than the "convex" location c if a constant feedrate V_0 is employed). Figure 12 shows the measured feedrate and (time–averaged) cutting force, as obtained from an experimental implementation of the PH curve interpolator for (24) on a CNC mill.

An important use of *time*–dependent feedrates $V(t)$ is the specification of smooth accelerations and decelerations along curved paths. This is especially important for high–speed machining, in which the dynamical issues of starting and stopping high–speed motions on curved paths become a serious concern. With G code part programs, for example, acceleration/deceleration intervals may span many short linear/circular segments, and thus require a cumbersome real–time block "look ahead" capability for their implementation.

Using PH curves, on the other hand, it is easy to specify a smooth feed acceleration from feedrate $V = 0$ for $t < 0$ up to a desired constant feedrate V_m for $t > T$ along a curve $\boldsymbol{r}(\xi)$. If $\tau = t/T$ is the "normalized" time during the acceleration interval $t \in [0, T]$, we use the polynomial feedrate function

$$V(\tau) = \sum_{k=0}^{n} V_k \binom{n}{k} (1-\tau)^{n-k} \tau^k \tag{25}$$

of *odd* degree n, with $V_0 = \cdots = V_{(n-1)/2} = 0$ and $V_{(n+1)/2} = \cdots = V_n = V_m$. This gives $C^{(n-1)/2}$ continuity with $V = 0$ for $t < 0$ and $V = V_m$ for $t > T$ — in particular, $n = 3$ and $n = 5$ yield C^1 and C^2 feedrate variations.

The interpolation equation for such time–dependent feedrate functions on PH curves is remarkably simple — it is precisely of the form (23), with the right–hand side being simply the integral $F(\tau)$ of (25), a polynomial of degree $n+1$. Only the right–hand side "constant" changes on using higher degrees n, and the incremental cost of evaluating this constant, in each sampling interval Δt, is insignificant for any "reasonable" choice of the degree n.

There are many other possibilites for specifying and optimizing feedrates along PH curves. For example, curved paths are realized on CNC machines by coordinated motions of independently–powered axes, and the chosen feedrates and feed accelerations should not impose demands on the motors of each axis that exceed their torque or power capacity. For PH curves, a thorough analysis

of this problem is possible [25], allowing an *a priori* determination of safe fixed feedrates and minimum feed acceleration intervals for a given path geometry.

§7. Motion from Curves II. Quaternion Methods

We have only been concerned, thus far, with the motion of *points* in *Euclidean spaces*. The problems of motion in non–Euclidean spaces, or motion of bodies of finite extent (involving changes of position *and* orientation), are much more challenging — they arise frequently in animation, robotics, 5–axis machining, dynamics, and many other applications. Thus, to conclude, we briefly consider some basic problems in the use of quaternions [5,8,52] to formulate spatial rigid body motions as time–parameterized loci in non–Euclidean spaces.

This subject has seen intense interest [17,27,35,37,40,48,54,59] — see also the extensive bibliography in [52] — in recent years, and substantial progress in "motion design" has been made. Nevertheless, the fundamental problems we encounter with purely translational (point) motion carry over to and, indeed, become much deeper in the context of general spatial motions.

The revival of quaternions in motion–design applications helps remedy a steady historical decline of interest in them. The introduction of quaternions by Sir William Hamilton (1805–1865) predates (and subsumes) development of the "ordinary" vector analysis in $\mathbb{R}^3$ by James Clerk Maxwell (1831–1879), Josiah Willard Gibbs (1839–1903), and Oliver Heaviside (1850–1925), who — along with later generations of physicists — considered quaternions to be an unduly cumbersome medium for describing the laws of nature [14].

A general displacement in Euclidean 3–space $\mathbb{R}^3$ can be interpreted as a screw displacement — i.e., a rotation about a fixed axis and a translation along that axis [8]. Six parameters are required to describe such displacements. Let $\boldsymbol{P} = (X, Y, Z)^T$ and $\boldsymbol{p} = (x, y, z)^T$ be point coordinates in a "fixed" frame Σ and a "movable" frame σ. The spatial displacement carrying Σ into σ may be described by an orthogonal rotation matrix $\boldsymbol{M}$ and a translation vector $\boldsymbol{d}$:

$$\boldsymbol{P} = \boldsymbol{M}\boldsymbol{p} + \boldsymbol{d}\,.$$

The rotation matrix can be expressed in terms of Euler parameters

$$c_0 = \cos\tfrac{1}{2}\phi\,, \quad c_1 = \lambda \sin\tfrac{1}{2}\phi\,, \quad c_2 = \mu \sin\tfrac{1}{2}\phi\,, \quad c_3 = \nu \sin\tfrac{1}{2}\phi,$$

where (λ, μ, ν) are direction cosines of the axis and ϕ is the rotation angle, as

$$\boldsymbol{M} = \begin{bmatrix} c_0^2 + c_1^2 - c_2^2 - c_3^2 & 2(c_1c_2 - c_0c_3) & 2(c_1c_3 + c_0c_2) \\ 2(c_2c_1 + c_0c_3) & c_0^2 - c_1^2 + c_2^2 - c_3^2 & 2(c_2c_3 - c_0c_1) \\ 2(c_3c_1 - c_0c_2) & 2(c_3c_2 + c_0c_1) & c_0^2 - c_1^2 - c_2^2 + c_3^2 \end{bmatrix}.$$

Note that, since the Euler parameters satisfy the normalization condition

$$c_0^2 + c_1^2 + c_2^2 + c_3^2 = 1\,, \tag{26}$$

only three are independent. In lieu of the translation vector $\boldsymbol{d} = (d_1, d_2, d_3)$, Study [56] introduced four new parameters:

$$\begin{aligned} c_0' &= (c_1 d_1 + c_2 d_2 + c_3 d_3)/2\,, \\ c_1' &= (-c_0 d_1 - c_3 d_2 + c_2 d_3)/2\,, \\ c_2' &= (c_3 d_1 - c_0 d_2 - c_1 d_3)/2\,, \\ c_3' &= (-c_2 d_1 + c_1 d_2 - c_0 d_3)/2\,, \end{aligned}$$

which, by definition, satisfy the constraint

$$c_0 c_0' + c_1 c_1' + c_2 c_2' + c_3 c_3' = 0\,. \tag{27}$$

The set of all displacements in $\mathbb{R}^3$ can then be regarded as points, or soma, in a 6–dimensional space spanned by eight coordinates $(c_0, c_1, c_2, c_3, c_0', c_1', c_2', c_3')$ subject to the two algebraic constraints (26) and (27).

A compact and elegant algebraic description of these soma is obtained [8] by combining the eight coordinates into a dual quaternion of the form

$$\mathcal{C} = c_0 + \varepsilon c_0' + \mathrm{i}(c_1 + \varepsilon c_1') + \mathrm{j}(c_2 + \varepsilon c_2') + \mathrm{k}(c_3 + \varepsilon c_3')\,. \tag{28}$$

Here, the quaternion basis elements satisfy the multiplication rules

$$\mathrm{ij} = \mathrm{k}, \quad \mathrm{jk} = \mathrm{i}, \quad \mathrm{ki} = \mathrm{j}, \quad \mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = -1$$

(so that multiplication is non–commutative: $\mathrm{ji} = -\mathrm{ij}$, etc). The components of the quaternion (28) are "dual numbers" of the form $x + \varepsilon\, x'$ for real x, x' — the dual basis element ε satisfies $\varepsilon^2 = 0$ $(\neq \varepsilon)$. The relations (26), (27) and $\varepsilon^2 = 0$ ensure that (28) is a *unit* dual quaternion: its components satisfy

$$(c_0 + \varepsilon c_0')^2 + (c_1 + \varepsilon c_1')^2 + (c_2 + \varepsilon c_2')^2 + (c_3 + \varepsilon c_3')^2 = 1\,.$$

For a spatial displacement specified by the unit dual quaternion (28), we can extract the geometrical parameters as follows:

$$\boldsymbol{d} = 2\,(c_0' \boldsymbol{c} - c_0 \boldsymbol{c}' - \boldsymbol{c} \times \boldsymbol{c}')\,, \quad \phi = 2\cos^{-1} c_0\,, \quad (\lambda, \mu, \nu) = \frac{\boldsymbol{c}}{\sin \frac{1}{2}\phi}\,,$$

where $\boldsymbol{c} = (c_1, c_2, c_3)$, $\boldsymbol{c}' = (c_1', c_2', c_3')$, $0 \le \cos^{-1} c_0 \le \pi$, and $\times$ is the familiar vector cross product in $\mathbb{R}^3$. Note also that the translation distance is

$$d = |\boldsymbol{d}| = 2\sqrt{c_0'^2 + c_1'^2 + c_2'^2 + c_3'^2}\,.$$

When $c_0' = c_1' = c_2' = c_3' = 0$ the dual quaternion (28) specifies a pure rotation (the non–commutativity of quaternion multiplication reflects the importance of the *order* in which spatial rotations are executed — see Figure 13).

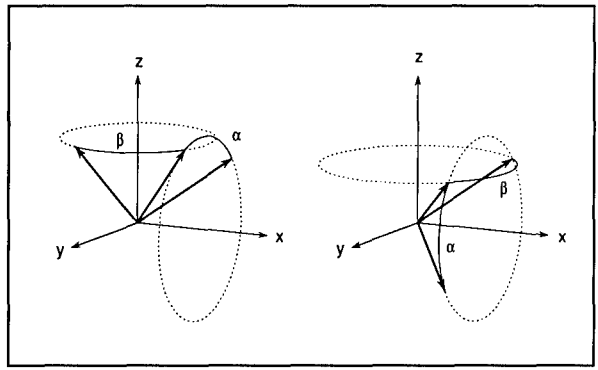


Fig. 13. Effect of rotations $R_x(\alpha)R_z(\beta)$ and $R_z(\beta)R_x(\alpha)$ applied to a vector $\boldsymbol{v}$.

The principal advantage of the form (28) is that the outcome of successive displacements, $\mathcal{A}$ followed by $\mathcal{B}$, corresponds to the ordered product $\mathcal{C} = \mathcal{B}\mathcal{A}$ of their dual quaternion representations. With

$$\mathcal{A} = a_0 + \varepsilon a_0' + \mathrm{i}\,(a_1 + \varepsilon a_1') + \mathrm{j}\,(a_2 + \varepsilon a_2') + \mathrm{k}\,(a_3 + \varepsilon a_3')\,,$$
$$\mathcal{B} = b_0 + \varepsilon b_0' + \mathrm{i}\,(b_1 + \varepsilon b_1') + \mathrm{j}\,(b_2 + \varepsilon b_2') + \mathrm{k}\,(b_3 + \varepsilon b_3')\,,$$

the elements of $\mathcal{C}$ are homogeneous quadratic forms in those of $\mathcal{A}$ and $\mathcal{B}$:

$$\begin{aligned} c_0 &= a_0b_0 - a_1b_1 - a_2b_2 - a_3b_3\,,\\ c_1 &= a_1b_0 + a_0b_1 + a_3b_2 - a_2b_3\,,\\ c_2 &= a_2b_0 + a_0b_2 + a_1b_3 - a_3b_1\,,\\ c_3 &= a_3b_0 + a_0b_3 + a_2b_1 - a_1b_2\,, \end{aligned}$$

$$\begin{aligned} c_0' &= a_0b_0' + a_0'b_0 - a_1b_1' - a_1'b_1 - a_2b_2' - a_2'b_2 - a_3b_3' - a_3'b_3\,,\\ c_1' &= a_1b_0' + a_1'b_0 + a_0b_1' + a_0'b_1 + a_3b_2' + a_3'b_2 - a_2b_3' - a_2'b_3\,,\\ c_2' &= a_2b_0' + a_2'b_0 + a_0b_2' + a_0'b_2 + a_1b_3' + a_1'b_3 - a_3b_1' - a_3'b_1\,,\\ c_3' &= a_3b_0' + a_3'b_0 + a_0b_3' + a_0'b_3 + a_2b_1' + a_2'b_1 - a_1b_2' - a_1'b_2\,. \end{aligned}$$

Now a general rigid body motion, involving both translation and rotation, corresponds to a locus of points in soma space, where a value of the time t is associated with each point. It is tempting to invoke a unit dual quaternion $\mathcal{C}(t)$ parameterized explicitly by "simple" — i.e., (piecewise) polynomial or rational — functions of time, and most motion design schemes rely upon this model. Typically, a sequence $\mathcal{C}_1, \ldots, \mathcal{C}_N$ of displacements at times $t_1, \ldots, t_N$ are specified, and one seeks a "smooth" motion interpolating them.

In §2 we emphasized that motion specification is as much concerned with velocities and accelerations — determined by the precise nature of the time parameterization — as with the path geometry, and this precept also holds for spatial rigid body motions. An *ad hoc* or "indirect" time parameterization of a quaternion locus $\mathcal{C}(t)$ can incur linear/angular velocities or accelerations that

are undesirable or, at least, only determinable *a posteriori* — after the motion has been specified, rather than being an integral part of its specification.

For rational point motion, the difficulty in simultaneously specifying both path geometry *and* speed along the path arises from the fact that curves do not, in general, admit rational arc length representations. As indicated in §6, the use of special (PH) curves can resolve this problem. For rational rigid body motions, however, the difficulty is not just one of computation, but also the more fundamental issue of how we characterize "distance travelled" in terms of both the translational and orientational components. Should they be treated together, or separately? In other words, can we introduce a suitable metric for soma space that allows us to define "arc length" along a unit dual quaternion locus $\mathcal{C}(t)$, and thus formulate methods to specify both positions/orientations *and* linear/angular speeds for rational spatial motions?

Ravani and Roth [49] have proposed, by analogy with elliptic geometry, a metric that yields a *dual number* value for distances in soma space. However, the use of this metric in motion design, or of alternate (real–valued) functions that exhibit the usual properties of metrics, remains to be explored.

§8. Closure

With its opening AXIOMS, or LAWS OF MOTION, the *Principia* establishes uniform motion as the natural state of a free body. The forces that act upon bodies incur deviations from uniform motion, in a deterministic though subtle manner that reveals appealing and useful connections between geometry and kinematics. Although, in this rather brief and eclectic survey, we have offered only a few illustrative anecdotes on this theme, we hope they have stirred the interest of some inquisitive readers, and have thus helped to promote further theoretical developments and practical applications.

References

1. Standard RS–274–D, Interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines, Electronic Industries Association, Washington, D.C., 1979.
2. Arnol'd, V. I., *Huygens and Barrow, Newton and Hooke* (translated by E. J. F. Primrose), Birkhäuser, Basel, 1990.
3. Bell, E. T., *Men of Mathematics*, Simon and Schuster, New York, 1937.
4. Beyer, R., *The Kinematic Synthesis of Mechanisms* (translated by H. Kuenzel), McGraw Hill, New York, 1963.
5. Blaschke, W., *Kinematik und Quaternionen*, Deutscher Verlag der Wissenschaften, Berlin, 1960.
6. Blechschmidt, J. L. and J. J. Uicker, Linkage synthesis using algebraic curves, ASME J. Mech. Trans. Auto. Design **108** (1986), 543–548.
7. Bohlin, K., Note sur le problème des deux corps et sur une intégration nouvelle dans le problème des trois corps, Bulletin Astronomique **28** (1911), 113–119.

8. Bottema, O. and B. Roth, *Theoretical Kinematics*, North Holland, 1979.
9. Boyer, C. B. and U. C. Merzbach, *A History of Mathematics*, Wiley, New York, 1991.
10. Chandrasekhar, S. *Newton's Principia for the Common Reader*, Oxford Univ. Press, 1995.
11. Chou, J–J. and D. C. H. Yang, Command axis generation for three–axis CNC machining, ASME J. Eng. Indus. **113** (August 1991), 305–310.
12. Cohen, I. B., *Introduction to Newton's Principia*, Harvard Univ. Press, 1971.
13. Cook, A., *Edmond Halley: Charting the Heavens and the Seas*, Oxford Univ. Press, 1998.
14. Crowe, M. J., *A History of Vector Analysis*, Dover, New York, 1994.
15. Descartes, R., *The Geometry of René Descartes* (translated from French and Latin by D. E. Smith and M. L. Latham), Dover, New York, 1954.
16. Erdman, A. G. and G. N. Sandor, *Mechanism Design* Vols. 1 and 2, Prentice–Hall, Englewood Cliffs, NJ, 1984.
17. Fang, Y. C., C. C. Hsieh, M. J. Kim, J. J. Chang, and T. C. Woo, Real time motion fairing with unit quaternions, Comput. Aided Design **30** (1998), 191–198.
18. Farouki, R. T., Optimal parameterizations, Comput. Aided Geom. Design **14** (1997), 153–168.
19. Farouki, R. T., Convergent inversion approximations for polynomials in Bernstein form, Comput. Aided Geom. Design **17** (2000), 179–196.
20. Farouki, R. T., J. Manjunathaiah, D. Nicholas, G–F. Yuan, and S. Jee, Variable–feedrate CNC interpolators for constant material removal rates along Pythagorean–hodograph curves, Comput. Aided Design **30** (1998), 631–640.
21. Farouki, R. T. and J. Rampersad, Cycles upon cycles: an anecdotal history of higher curves in science and engineering, in *Mathematical Methods for Curves and Surfaces II* (M. Daehlen, T. Lyche, and L. L. Schumaker, eds.), Vanderbilt Univ. Press, Nashville TN (1998), 95–116.
22. Farouki, R. T. and T. Sakkalis, Pythagorean hodographs, IBM J. Res. Develop. **34** (1990), 736–752.
23. Farouki, R. T. and T. Sakkalis, Real rational curves are not "unit speed," Comput. Aided Geom. Design **8** (1991), 151–157.
24. Farouki, R. T. and S. Shah, Real–time CNC interpolators for Pythagorean hodograph curves, Comput. Aided Geom. Design **13** (1996), 583–600.
25. Farouki, R. T., Y–F. Tsai, and C. R. Wilson, Feedrate and feed acceleration constraints along curved tool paths, Comput. Aided Geom. Design (1999), submitted.
26. Galilei, G. (translated by S. Drake), *Dialogue Concerning the Two Chief World Systems* (2nd ed.), Univ. of California Press, 1967.

27. Ge, Q. J. and B. Ravani, Geometric construction of Bézier motions, ASME J. Mech. Design **116** (1995), 749–755.

28. Goldstein, H., *Classical Mechanics* (2nd ed.), Addison–Wesley, Reading MA, 1980.

29. Grashof, F., *Theoretische Maschinenlehre* (3 vols.), Voss, Leipzig, 1883.

30. Hartenberg, R. S. and J. Denavit, *Kinematic Synthesis of Linkages*, McGraw Hill, New York, 1964.

31. Hofmann, J. E., *Leibniz in Paris 1672–1676: His Growth to Mathematical Maturity*, Cambridge Univ. Press, 1974.

32. Hrones, J. A. and G. L. Nelson, *Analysis of the Four–Bar Linkage: Its Application to the Synthesis of Mechanisms*, The Technology Press of MIT, Cambridge, MA, and Wiley, New York, 1951.

33. Hunt, K. R., *Kinematic Geometry of Mechanisms*, Oxford Univ. Press, 1990.

34. Huygens, C., *Horologium Oscillatorium sive De Motu Pendulorum ad Horologia Aptato Demonstrationes Geometricæ*, Paris, 1673; translated into English by R. J. Blackwell as *The Pendulum Clock, or Geometrical Demonstrations Concerning the Motion of Pendula as Applied to Clocks*, Iowa State Univ. Press, Ames, IA, 1986.

35. Jüttler, B., Visualization of moving objects using dual quaternion curves, Computers and Graphics **18** (1994), 315–326.

36. Jüttler, B., A vegetarian approach to optimal parameterizations, Comput. Aided Geom. Design **14** (1997), 887–890.

37. Jüttler, B. and Wagner M., Computer aided design with spatial rational B–spline motions, ASME J. Mech. Design **118** (1996), 193–201.

38. Kasner, E., *Differential–Geometric Aspects of Dynamics*, American Math. Society, New York, 1913.

39. Kempe, A. B., On a general method of describing plane curves of the n–th degree by linkwork, Proc. London Math. Soc. **7** (1876), 213–216.

40. Kim, M.–S. and K.–W. Nam, Interpolating solid orientations with circular blending quaternion curves, Comput. Aided Design **27** (1995), 385–398.

41. Lawden, D. F., *Elliptic Functions and Applications*, Springer, New York, 1989.

42. Minkowski, H., Space and Time, translation of an address delivered at the 80th Assembly of German Natural Scientists and Physicians, Sept. 21, 1908, Köln, in *The Principle of Relativity*, Dover, 1952.

43. Moon, H. P., Minkowski Pythagorean hodographs, Comput. Aided Geom. Design **16** (1999), 739–753.

44. Needham, T., Newton and the transmutation of force, American Math. Monthly **100** (1993), 119–137.

45. Needham, T., *Visual Complex Analysis*, Oxford Univ. Press, 1997.

46. Newton, I., *Philosophiæ Naturalis Principia Mathematica*, 1686; Florian Cajori's revision of Andrew Motte's translation, published as *Sir Isaac Newton's Mathematical Principles of Natural Philosophy and His System of the World*, Vols. 1 & 2, Univ. of California Press, Berkeley, 1962.

47. Nolle, H., Linkage coupler curve synthesis: A historical review I. Developments up to 1875 & II. Developments after 1875, Mech. Mach. Theory **9** (1974), 147–168 & 325–348.

48. Park, F. C. and B. Ravani, Smooth invariant interpolation of rotations, ACM Trans. Graphics **16** (1997), 277–295.

49. Ravani, B. and B. Roth, Mappings of spatial kinematics, ASME J. Mech. Transm. Auto. Design **106** (1984), 341–347.

50. Roberts, S., Three–bar motion in plane space, Proc. London Math. Soc. **7** (1875), 14–23.

51. Rogers, P. (ed.), *Alexander Pope: A Critical Edition of the Major Works*, Oxford Univ. Press, 1993.

52. Röschel, O., Rational motion design — a survey, Comput. Aided Design **30** (1998), 169–178.

53. Sakkalis, T. and R. Farouki, Algebraically rectifiable parametric curves, Comput. Aided Geom. Design **10** (1993), 551–569.

54. Shoemake, K., Animating rotation with quaternion curves, Computer Graphics **19** (1985), 245–254.

55. Stillwell, J., *Mathematics and Its History*, Springer, New York, 1989.

56. Study, E., *Die Geometrie der Dynamen*, Leipzig, 1903.

57. Symon, K. R., *Mechanics* (3rd ed.), Addison–Wesley, Reading MA, 1971.

58. Wampler, C. W., A. P. Morgan, and A. J. Sommese, Complete solution of the nine–point path synthesis problem for four–bar linkages, *Mechanism Synthesis and Analysis* DE–Vol. 25 (Proceedings of 1990 ASME Design Technical Conference, Chicago), 361–368.

59. Wang, W. and B. Joe, Orientation interpolation in quaternion space using spherical biarcs, Proceedings, Graphics Interface '93, 24–32.

60. Westfall, R. S., *Force in Newton's Physics: The Science of Dynamics in the Seventeenth Century*, American Elsevier, New York, 1971.

61. Yang, D. C. H. and T. Kong, Parametric interpolator versus linear interpolator for precision CNC machining, Comput. Aided Design **26** (1994), 225–234.

Rida T. Farouki
Department of Mechanical and Aeronautical Engineering,
University of California, Davis, CA 95616, USA.
farouki@ucdavis.edu

Curvature and Tangency Handles for Control of Convex Cubic Shapes

G. Figueroa, M. Paluszny, and F. Tovar

Abstract. We consider the problem of modelling a plane convex shape with a closed component of an algebraic non-singular cubic. All nine degrees of freedom are interpreted as visual handles, namely: tangency to three prescribed lines at three given points and the curvatures at these points.

§1. Introduction

Algebraic curves, beyond conics, where introduced in CAGD by Sederberg [6] in 1984. Paluszny and Patterson [1–3] studied splines constructed with segments of cubic algebraic curves, called A-splines. Tovar, Paluszny and Patterson [4] looked at A-splines constructed with segments of singular algebraic cubics, which are just rational cubics, with new, geometrically more meaningful, control handles for their shape. Non-singular cubics are classically well-known objects Salmon [5]. Projectively they could be of two types: two or one circuit cubics. A two circuit cubic consists of two pieces, one of which can be realized affinely as a convex closed curve, called an oval.

The goal of this paper is to study the feasibility of using ovals to model plane convex shapes. We remark that ovals are not splines, they are C^∞ curves. Therefore, the main advantage of using ovals for modelling of convex shapes is that no stitching of segments is required, which would be the case for splines. In fact, a convex shape represented by a cubic oval doesn't have any joints at all. It seems natural to place the study of cubic ovals within the context of A-splines because they are connected components of cubic algebraic curves. Moreover, the techniques to study the shape handles for their control are similar to those used for A-splines as mentioned above. In particular, we want to control the shape of an oval through its control triangle, contact interpolation, and curvatures at three prescribed points, see Figures 1–4.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 91–98.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

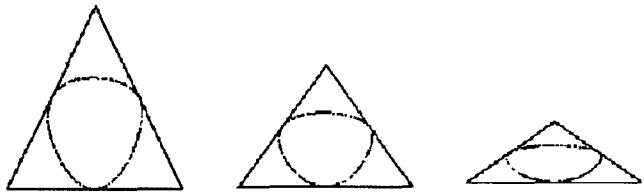


Fig. 1. Shape control by moving vertices of the control triangle.

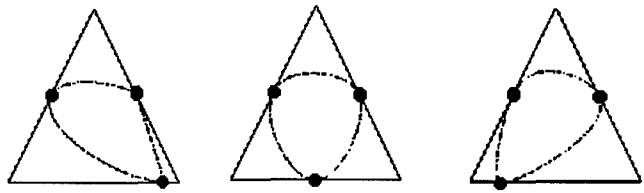


Fig. 2. Shape control by moving one of the tangency points.

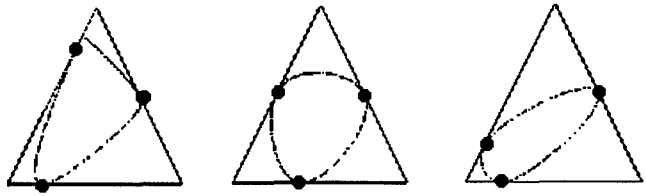


Fig. 3. Shape control by moving two of the tangency points.

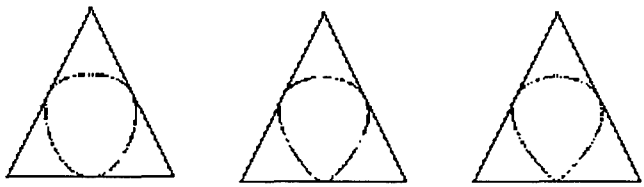


Fig. 4. Sharpening the curvature at one contact point.

§2. Barycentric Coordinates and Curvature at the Endpoints

The general algebraic cubic in cartesian coordinates x, y is given by

$$F(x,y) = a_{30}x^3 + a_{03}y^3 + a_{21}x^2y + a_{12}xy^2 + a_{20}x^2 + a_{02}y^2 + a_{11}xy + a_{10}x + a_{01}y + a_{00}.$$

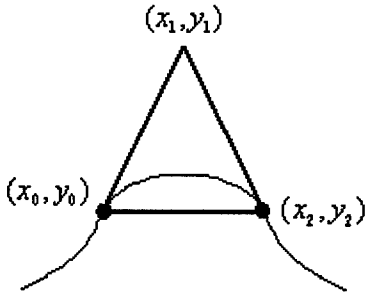


Fig. 5. Cubic with two prescribed tangencies.

If the cubic interpolates two points (x_0, y_0) and (x_2, y_2), and it is tangent to two lines joining these points to a third point (x_1, y_1) as shown in Figure 5, then its expression in terms of the barycentric coordinates (S, T, U) with respect to the triangle of vertices (x_0, y_0), (x_1, y_1) and (x_2, y_2) reduces to

$$F(S,T,U) = aS^2U + bSU^2 - cST^2 - dT^2U + eSTU + fT^3 \qquad (1)$$

where a, b, c, d, e and f are arbitrary real coefficients. This was observed by Sederberg [6]. In Paluszny and Patterson [2] it was shown that the curvatures $\bar{k}_s$ and $\bar{k}_u$ of (1) at (x_0, y_0) and (x_2, y_2) are given by

$$\bar{k}_u = \frac{c}{a} \frac{\Delta}{(\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2})^3},$$

$$\bar{k}_s = \frac{d}{b} \frac{\Delta}{(\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2})^3},$$

where Δ is the area of the triangle with vertices (x_0, y_0), (x_1, y_1) and (x_2, y_2).

§3. Three Prescribed Contacts

We now focus on the family of cubics with three prescribed contacts. Consider a triangle of vertices P_0, P_1 and P_2, and let (s, t, u) be the corresponding barycentric coordinates. Choose one point on each side, as shown in Figure 6.

The barycentric coordinates of each Q_i are $Q_0 = Q_0(0, t_0, u_0)$, $Q_1 = Q_1(s_1, 0, u_1)$, and $Q_2 = Q_2(s_2, t_2, 0)$. The general equation of a cubic passing through these points and tangent (or singular) at them to the sides of the

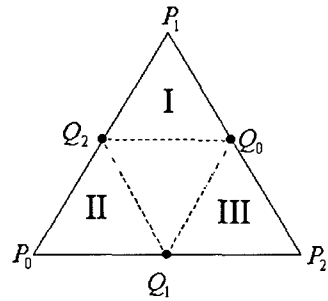


Fig. 6. Barycentric coodinates of side points.

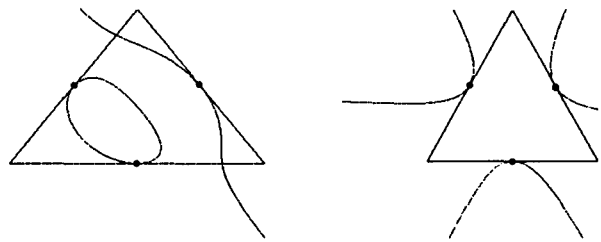


Fig. 7. Wrong contacts.

triangle is

$$
\begin{aligned}
G(s,t,u) = {} & a_{300}s^3 + a_{030}t^3 + a_{003}u^3 + \frac{(s_2^3 a_{300} - 2a_{030}t_2^3)st^2}{s_2 t_2^2} \\
+ {} & \frac{(a_{030}t_2^3 - 2s_2^3 a_{300})s^2 t}{s_2^2 t_2} + \frac{(t_0^3 a_{030} - 2a_{003}u_0^3)tu^2}{t_0 u_0^2} \\
+ {} & \frac{(a_{003}u_0^3 - 2t_0^3 a_{030})t^2 u}{t_0^2 u_0} + \frac{(s_1^3 a_{300} - 2a_{003}u_1^3)su^2}{s_1 u_1^2} \\
+ {} & \frac{(a_{003}u_1^3 - 2s_1^3 a_{300})s^2 u}{s_1^2 u_1} + a_{111}stu.
\end{aligned}
$$

Note that there are four free homogeneous parameters. These have to be used for two purposes:

- to guarantee that the cubic is actually a two circuit cubic and that the interpolation occurs at points of the oval,
- to express the free parameters in terms of the curvatures at the interpolation points, and find interval ranges for their meaningful modification. The first point is crucial because we need to preclude situations in which the contacts occur at points off the oval, as illustrated in Figure 7.

§4. Curvatures at the Contact Points

We need to control the curvatures at the interpolation points. The first step will be to find formulas for the curvatures in terms of the free parameters a_{300}, a_{030}, a_{003} and a_{111} of the cubic, as expressed in the barycentric coordinates (s,t,u). In the next section we will produce inversion formulas for the a_{ijk} in terms of the curvatures, which will allow us to find oval shapes with prescribed curvatures at the prescribed contacts. It will be convenient to express the curvatures $\bar{k}_s$ and $\bar{k}_u$ at Q_2 and Q_0 respectively, in terms of the barycentric coordinates s,t,u with respect to P_0, P_1 and P_2. As remarked in the previous section

$$\bar{k}_u = \frac{c_I}{a_I}\frac{\Delta_I}{(\overline{P_1Q_0})^3},$$

where Δ_I is the area of triangle I, $\overline{P_1Q_0}$ denotes the distance between P_1 and Q_0, and a_I, b_I, c_I and d_I, are the coefficients of the cubic in the barycentric coordinates with respect to Q_2, P_1 and Q_0, compare with (1). And similarly

$$\bar{k}_s = \frac{d_I}{b_I}\frac{\Delta_I}{(\overline{P_1Q_2})^3}.$$

In fact if a,b,c,d,e,f are the coefficients of the cubic in the barycentric coordinates with respect to the triangle $P_0P_1P_2$ as expressed in (1), then the coefficients a_I, b_I, c_I and d_I of this cubic with respect to the triangle I of vertices Q_2, P_1 and Q_0 are obtained by a linear change of variables:

$$a_I = \frac{u_0u_1(\alpha^2+\beta^2)a_{003} - t_0^2s_1^2(2a_{300}u_0s_2^2s_1 + 2t_0a_{030}t_2^2u_1 - a_{111}s_2t_2u_0u_1)}{s_1^2u_1t_0^2},$$

$$b_I = \frac{s_2s_1(\alpha^2+\beta^2)a_{300} - t_2^2u_1^2(2a_{030}t_0^2s_1t_2 + 2s_2u_0^2a_{003}u_1 - a_{111}s_2t_0u_0s_1)}{t_2^2s_1u_1^2},$$

where $\alpha = u_1s_2t_0$, $\beta = u_0t_2s_1$, and

$$c_I = -\frac{a_{030}t_2^3 + s_2^3a_{300}}{t_2^2},$$

$$d_I = -\frac{a_{003}u_0^3 + a_{030}t_0^3}{t_0^2}.$$

Using the relationship between the areas of the triangles $P_0P_1P_2$ and $Q_2P_1Q_0$ as illustrated in Figure 6, it follows that $\bar{k}_u$ and $\bar{k}_s$ can be expressed directly in terms of the geometry of the triangle $P_0P_1P_2$:

$$\bar{k}_u = \frac{4c_I}{a_I}\frac{u_0}{s_2^2}\frac{\Delta}{(\overline{P_0P_1})^3}, \tag{2}$$

$$\bar{k}_s = \frac{4d_I}{b_I}\frac{s_2}{u_0^2}\frac{\Delta}{(\overline{P_1P_2})^3}, \tag{3}$$

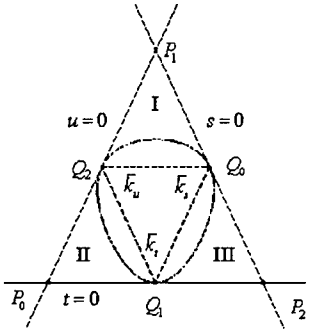


Fig. 8. Curvatures at the contact points.

where Δ is the area of the triangle $P_0P_1P_2$. To compute the curvature $\bar{k}_t$ at Q_1 with barycentric coordinates $(s_1, 0, u_1)$, we consider triangle II to get

$$\bar{k}_t = \frac{4d_{II}}{b_{II}} \frac{t_2}{u_1^2} \frac{\Delta}{(\overline{P_0P_2})^3}. \tag{4}$$

It is easy to show that using triangle III instead, we obtain the same expression for $\bar{k}_t$.

Figure 8 illustrates the relationships between the coordinates of contact points Q_0, Q_1 and Q_2 and curvatures k_s, k_t and k_u.

§5. Inversion Formulas and Shape Control

Equations (2)–(4) can be rewritten as

$$k_u s_2^2 a_I - u_0 c_I = 0$$

$$k_s u_0^2 b_I - s_0 d_I = 0$$

$$k_t u_1^2 a_{II} - t_2 c_{II} = 0,$$

where the a, b, c, d coefficients are given in terms of a_{300}, a_{030}, a_{003}, and a_{111} and k_s, k_t and k_u are proportional to the curvatures $\bar{k}_s$, $\bar{k}_t$ and $\bar{k}_u$, i.e.

$$k_u = \bar{k}_u \frac{\overline{P_0P_1}^3}{\Delta}, \qquad k_t = \bar{k}_t \frac{\overline{P_0P_2}^3}{\Delta}, \qquad k_s = \bar{k}_s \frac{\overline{P_1P_2}^3}{\Delta}.$$

The formulas for the a_{ijk} in terms of k_s, k_t and k_u are

$$\begin{aligned} a_{300} = {} & u_1^2 t_2^2 (k_t k_u (\alpha+\beta)^3 (s_1 s_2 - t_0 u_0 (\alpha+\beta) k_s) \\ & + u_1 t_2 (t_0^3 u_0^3 k_s - s_2^3 t_0^3 k_u - s_1^3 u_0^3 k_t)), \end{aligned}$$

$$\begin{aligned} a_{030} = {} & u_0^2 s_2^2 (k_s k_u (\alpha+\beta)^3 (t_0 t_2 - s_1 u_1 (\alpha+\beta) k_t) \\ & + s_2 u_0 (u_1^3 s_1^3 k_t - u_1^3 t_0^3 k_s - s_1^3 t_2^3 k_u)), \end{aligned}$$

$$a_{003} = t_0^2 s_1^2 (k_s k_t (\alpha+\beta)^3 (u_0 u_1 - t_2 s_2 (\alpha+\beta) k_u) + s_1 t_0 (s_2^3 t_2^3 k_u - t_2^3 u_0^3 k_s - s_2^3 u_1^3 k_t)), \tag{5}$$

$$\begin{aligned} a_{111} = {} & (\alpha+\beta)^3 (k_s k_t k_u (\alpha+\beta)((\alpha-\beta)^2 - 2\alpha\beta) \\ & + 2k_s k_t u_0^2 t_0 s_1 u_1^2 + 2k_s k_u u_0 t_0 t_2^2 s_2 + 2t_2 s_2^2 s_1^2 u_1^2 k_t k_u) \\ & - 3(\alpha^2+\beta^2)(t_0 t_2^2 s_2^2 s_1 k_u + u_0^2 t_0^2 t_2 u_1 k_s + u_0 s_2 s_1^2 u_1^2 k_t) \\ & + (\alpha+\beta)((\alpha-\beta)^2 + \alpha\beta). \end{aligned}$$

As established above, see Figure 7, further constraints are required for the contacts to occur at points of the oval, and for the latter to be contained inside the triangle $P_0P_1P_2$. Given the points Q_0, Q_1 and Q_2, (2) determines a cubic which is tangent to the sides of triangle $P_0P_1P_2$ at these points. To guarantee that the points Q_0, Q_1 and Q_2 lie on the oval it is sufficient that the coefficients a_{300}, a_{030} and a_{003} have the same sign. Indeed, the first condition implies that the third intersection of the cubic with each of the lines $s = 0$, $t = 0$ and $u = 0$ occurs outside the triangle. Since each contact Q_i accounts for two intersections, if the cubic at every Q_i bends inwards the triangle $P_0P_1P_2$, then the contacts actually occur at points of the oval of the cubic which then has to lie inside the triangle. The positivity of k_s, k_t and k_u guarantees that the curve bends towards the interior of the triangle, see [3].

Given the contacts at $Q_0(0, t_0, u_0)$, $Q_1(s_1, 0, u_1)$ and $Q_2(s_2, t_2, 0)$, see Figure 6, for a_{003}, a_{030} and a_{003} to have the same sign, it is enough to take

$$k_s^0 = \frac{s_2 s_1}{t_0 u_0 (\alpha+\beta)},$$

$$k_t^0 = \frac{t_0 t_2}{u_1 s_1 (\alpha+\beta)}, \tag{6}$$

$$k_u^0 = \frac{u_0 u_1}{s_2 t_2 (\alpha+\beta)}.$$

Note that given an oval with three prescribed curvatures, when we fix two of them the equations for the a_{ijk} in terms of the third are linear. So, it is easy to find an interval range for the modification of the third curvature, while keeping the coefficients a_{300}, a_{030} and a_{003} of the same sign.

§6. Conclusion

A convex shape can be modelled with a cubic oval controlled by a triangle that contains it. By moving the vertices, we drag the oval along (see Figure 1). Moreover, given any triangle and three points, one on each of its sides, we can produce a convex shape that contacts the triangle at these points. This convex shape is our cubic oval. Furthermore, we can prescribe the curvatures at these three points, within precisely defined conditions. If we fix two curvatures, the third curvature can be modified within an interval which can be computed by

solving a system of linear inequalities. In fact given the vertices of the control triangle, the contact points at its sides and the three initial curvatures k_s^0, k_t^0 and k_u^0 as given by (6), it is always possible to modify any of them keeping the other two fixed.

This entails solving the linear inequalities $a_{300} > 0$, $a_{030} > 0$ and $a_{003} > 0$, or $a_{300} < 0$, $a_{030} < 0$ and $a_{003} < 0$ given by the linear system (5).

References

1. Paluszny, M. and R. Patterson, A family of tangent continuous cubic algebraic curves. ACM Transactions on Graphics 12 (1993), 209–232.
2. Paluszny, M. and R. Patterson, Curvas Algebraicas de Grado Tres en Modelación Geométrica. V Escuela Venezolana de Matemática (1992).
3. Paluszny, M. and R. Patterson, Geometric control of G^2-cubic A-splines. Comput. Aided Geom. Design **15** (1998), 261–287.
4. Paluszny, M., R. Patterson, and F. Tovar, G^2-composite cubic Bézier curves. Journal of Computational and Applied Mathematics 102 (1999), 49–71.
5. Salmon, G. *Higher Plane Curves*, Hodges, Foster and Figgis, Dublin, 1879.
6. Sederberg, T. Planar piecewise algebraic curves, Comput. Aided Geom. Design **1** (1984), 241–255.

G. Figueroa[1], M. Paluszny[2] and F. Tovar[3]
Universidad Central de Venezuela. Facultad de Ciencias.
Escuela de Matemática.
Laboratorio de Computación Gráfica y Geometría Aplicada.
Apartado 47809. Los Chaguaramos. Caracas 1041-A. Venezuela.
[1] giovanni@euler.ciens.ucv.ve
[2] marco@euler.ciens.ucv.ve
[3] ftovar@euler.ciens.ucv.ve

Monotonicity Conditions of Curvature for Bézier-de Casteljau Curves

Jean-Charles Fiorot and Laurent Schiavon

Abstract. In this paper, we deal with the monotonicity of curvature problem for Bézier-de Casteljau curves. We focus more particularly on the cubic case. A condition about decreasing curvature at the origin of a cubic curve is given so that it implies decreasing curvature at every point. The corresponding cubics are determined by their control polygon.

§1. Introduction

The problem discussed here concerns the shape control of curves; mainly how to obtain curves with a monotone variation of curvature. The aim is to find the widest class of curves with monotonely increasing or decreasing curvature variation. This problem mainly arises in car body shape design.

Let P be a *n-degree* Bézier-de Casteljau planar curve defined on $[0,1]$ given by its control polygon $\{P_0, P_1, \ldots, P_n\}$. Let us consider $l = \mid P_0P_1 \mid$, and, $\forall i = 1, \ldots, n-1$, set $h_i = \mid P_iP_{i+1} \mid / \mid P_{i-1}P_i \mid$ and $\varphi_i = (P_iP_{i+1}, P_{i-1}P_i)$. We say that P admits the representation $(h_1, \ldots, h_{n-1}; \varphi_1, \ldots, \varphi_{n-1}; l)$. Higashi, Kaneko and Hosaka [3] characterized the monotonicity of curvature when $\forall k = 1, \ldots, n-1$, $h_k = h$ and $\varphi_k = \varphi$: $h\cos\varphi \geq 1$ and $h \leq \cos\varphi$ are respectively the condition of decrease and increase of the curvature. This model has been used recently by Mineur, Lychah, Castelain and Giaume [4] to control a shape when fitting a curve to a set of given data.

Here, we focus on the cubic case. From the representation $(h_1, h_2; \varphi_1, \varphi_2; l)$ of P, we apply the de Casteljau Algorithm at t value belonging to the interval of definition, and we determine by induction the parameters characterizing the two segments given by the Subdivision Algorithm. This process enables us to determine the curvature ρ. Then, we seek cubic curves for which decreasing curvature at the origin implies decreasing curvature at every point. Such curves are determined *via* the parameters $r^* = h_2/h_1$, φ_1 and φ_2. The study falls into two cases : $r^* \geq 1$ and $r^* < 1$. For the second case, we give a more

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 99–108.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

strict decreasing curvature condition at the origin in order to get the decrease of ρ everywhere.

In the framework of Fiorot and Jeannin [1] and Fiorot, Jeannin and Taleb [2], an attempt to extend these results to rational cubic curves has been made.

§2. de Casteljau Algorithm

Let P be a Bézier-de Casteljau cubic curve (BCc in short) defined on $[0,1]$ with its control polygon $\{P_0,P_1,P_2,P_3\}$ and representation $(h_1,h_2;\varphi_1,\varphi_2;l)$. For $t\in[0,1]$, the de Casteljau algorithm gives the points $(P_k^{(j)})_{k=0,\cdots,3-j}^{j=0,\cdots,3}$ defined by the relation $P_k^{(j+1)}=(1-t)P_k^{(j)}+tP_{k+1}^{(j)}$ and we obtain $P(t)=P_0^{(3)}$. Now, we set $\forall k=1,2$,

$$\gamma_k^{(1)}=\left(P_{k-1}^{(1)}P_{k-1}^{(2)},P_{k-1}P_{k-1}^{(1)}\right),\qquad \delta_k^{(1)}=\left(P_k^{(1)}P_{k+1},P_{k-1}^{(2)}P_k^{(1)}\right),$$

$$\Delta_k^{(0)}=\frac{|\,P_{k-1}^{(1)}P_{k-1}^{(2)}\,|}{|\,P_{k-1}P_{k-1}^{(1)}\,|},$$

and

$$h_1^{(1)}=\frac{|\,P_1^{(1)}P_2^{(1)}\,|}{|\,P_0^{(1)}P_1^{(1)}\,|},\quad \varphi_1^{(1)}=\left(P_1^{(1)}P_2^{(1)},P_0^{(1)}P_1^{(1)}\right),\quad \Delta_1^{(1)}=\frac{|\,P_0^{(2)}P_0^{(3)}\,|}{|\,P_0^{(1)}P_0^{(2)}\,|},$$

$$\gamma_1^{(2)}=\left(P_0^{(2)}P_0^{(3)},P_0^{(1)}P_0^{(2)}\right),\qquad \delta_1^{(2)}=\left(P_1^{(2)}P_2^{(1)},P_0^{(3)}P_1^{(2)}\right).$$

After some calculations, we obtain $\forall k=1,2$,

$$\Delta_k^{(0)}=\left((1-t)^2+2t(1-t)h_k\cos\varphi_k+(h_kt)^2\right)^{\frac{1}{2}}. \tag{1}$$

Then,

$$\exp(i\gamma_k^{(1)})=\left((1-t)+h_k\exp(i\varphi_k)t\right)/\Delta_k^{(0)}, \tag{2}$$

$$\exp(i\delta_k^{(1)})=\left((1-t)\exp(i\varphi_k)+h_kt\right)/\Delta_k^{(0)}, \tag{3}$$

$$\varphi_1^{(1)}=\gamma_2^{(1)}+\delta_1^{(1)}\quad[2\pi], \tag{4}$$

$$h_1^{(1)}=\frac{\Delta_2^{(0)}}{\Delta_1^{(0)}}h_1, \tag{5}$$

$$\Delta_1^{(1)}=\left((1-t)^2+2t(1-t)h_1^{(1)}\cos\varphi_1^{(1)}+(h_1^{(1)}t)^2\right)^{\frac{1}{2}}, \tag{6}$$

$$\exp(i\gamma_1^{(2)}) = \left((1-t) + h_1^{(1)}\exp(i\varphi_1^{(1)})t\right)/\Delta_1^{(1)}, \tag{7}$$

$$\exp(i\delta_1^{(2)}) = \left((1-t)\exp(i\varphi_1^{(1)}) + h_1^{(1)}t\right)/\Delta_1^{(1)}. \tag{8}$$

Remark 1. For $t \in [0,1]$, the de Casteljau Algorithm splits the curve P into two segments: the left one is the restriction of P to the interval $[0,t]$ defined by the control polygon $\{P_0\ ,\ P_0^{(1)},\ P_0^{(2)},\ P_0^{(3)}\}$ which admits the representation $(\Delta_1^{(0)}, \Delta_1^{(1)}; \gamma_1^{(1)}, \gamma_1^{(2)}; lt)$, whereas the right one is the restriction to $[t,1]$ defined by $\{P_0^{(3)},\ P_1^{(2)},\ P_2^{(1)},\ P_3\}$ which admits $(h_1^{(1)}/\Delta_1^{(1)}, h_2/\Delta_2^{(0)};\ \delta_1^{(2)}, \delta_2^{(1)};\ \Delta_1^{(0)}\Delta_1^{(1)}l(1-t))$ as a representation.

§3. Curvature Characterization

We determine the curvature radius R and the curvature $\rho = \frac{1}{|R|}$ at every point of the curve P *via* the parameters mentioned above. Let us remember that the curvature radius R is determined by the approximation $R \simeq \frac{\Delta s}{\Delta\alpha}$, where s denotes the arc length and α the angle of the tangent vector with a fixed direction.

First, we give the expression of R at $t = 0$. Let us consider a value t near 0. We have the following approximations

$$\Delta s \simeq (1 + \Delta_1^{(0)} + \Delta_1^{(0)}\Delta_1^{(1)})lt,$$
$$\Delta\alpha = -\gamma_1^{(1)} - \gamma_1^{(2)} \simeq -\sin(\gamma_1^{(1)} + \gamma_1^{(2)}).$$

By using (1) – (7), we obtain when t tends to 0

$$R(0) = -\frac{3}{2}\frac{l}{h_1 \sin\varphi_1}. \tag{9}$$

Now for $t \in (0,1]$, we apply (9) to the right segment whose origin is $P_0^{(3)}$. Then, the curvature radius at t is

$$R(t) = -\frac{3}{2}\frac{\Delta_1^{(0)}\Delta_1^{(1)}l(1-t)}{(h_1^{(1)}/\Delta_1^{(1)})\sin\delta_1^{(2)}}.$$

Therefore, (8) implies

$$R(t) = -\frac{3}{2}\frac{\Delta_1^{(0)}(\Delta_1^{(1)})^3 l}{h_1^{(1)}\sin\varphi_1^{(1)}}. \tag{10}$$

Application. Let us consider the case $h_1 = h_2 = h$ and $\varphi_1 = \varphi_2 = \varphi$ with $\varphi \in [0, \frac{\pi}{2}]$. One can prove that $\forall t \in [0,1]$, $\Delta_1^{(0)} = \Delta_1^{(1)} (= \Delta)$, $\varphi_1^{(1)} = \varphi$ and $h_1^{(1)} = h$. Then, we obtain with (10), $\forall t \in [0,1]$, $\rho(t) = 2h\sin\varphi/(3l\Delta^4)$.
Differentiating $\Delta = \left((1-t)^2 + 2t(1-t)h\cos\varphi + (ht)^2\right)^{\frac{1}{2}}$, we deduce that $h\cos\varphi \geq 1$ and $h \leq \cos\varphi$ are respectively the condition of decrease and increase of the curvature [3].

§4. Decreasing Curvature Condition in the Case $r^* \geq 1$

Let P be a BCc curve with representation $(h_1, h_2; \varphi_1, \varphi_2; l)$, and let $(\varphi_1, \varphi_2) \in [0, \frac{\pi}{2})^2$. For $t \in [0,1]$, we set $r = \Delta_1^{(1)}/\Delta_1^{(0)}$. At $t = 1$, this parameter is r^*. Moreover, we define

$$\lambda = \frac{\sin(\gamma_1^{(1)} + \gamma_1^{(2)})}{\sin 2\gamma_1^{(1)}}, \qquad \mu = \frac{\sin(\gamma_1^{(1)} + \gamma_1^{(2)})}{\sin 2\gamma_1^{(2)}}.$$

At $t = 1$, we prove *via* (1) – (7) that these parameters are respectively

$$\lambda^* = \frac{\sin(\varphi_1 + \varphi_2)}{\sin 2\varphi_1}, \qquad \mu^* = \frac{\sin(\varphi_1 + \varphi_2)}{\sin 2\varphi_2}.$$

Lemma 1. *At $t = 0$ and $t = 1$, we have respectively the equivalences :*

$$\frac{3}{2} h_1 \cos\varphi_1 \left(1 - \frac{r^*\lambda^*}{3}\right) \geq 1 \Longleftrightarrow \rho'(0) \leq 0, \tag{11}$$

$$h_2 \geq \frac{3}{2}\left(1 - \frac{r^*\mu^*}{3}\right)\cos\varphi_2 \Longleftrightarrow \rho'(1) \leq 0. \tag{12}$$

Proof: We calculate $\rho'(0)$ as $\lim_{t\to 0}(\rho(t) - \rho(0))/t$ *via* (9) and (10). We obtain

$$\rho'(0) = \rho(0)(h_1(h_2 \sin(\varphi_1 + \varphi_2) - 2\sin\varphi_1 - 6\sin\varphi_1(h_1\cos\varphi_1 - 1)). \tag{13}$$

Then, with the above definitions of λ^* and μ^*, we obtain (11). At $t = 1$, we consider the curve $P(1-t)$ which has the representation $(1/h_2, 1/h_1; \varphi_2, \varphi_1; h_1 h_2 l)$. □

Remark 2. The equivalences (11) and (12) do not depend on the interval of definition.

Lemma 2. *Let P be a BCc curve defined on interval $[t_1, t_2]$ with the representation $(h_1, h_2; \varphi_1, \varphi_2; l)$. Let us suppose that $r^* \geq 1$. Then,*

$$\rho'(t_1) \leq 0 \Rightarrow \rho'(t_2) \leq 0.$$

Proof: We have the successive inequalities

$$h_2 \geq h_1 \geq \frac{2}{3}\left(1 - \frac{r^*\lambda^*}{3}\right)^{-1} \frac{1}{\cos\varphi_1} \geq \frac{3}{2}\left(1 - \frac{r^*\mu^*}{3}\right)\cos\varphi_2.$$

The first one uses the definition of r^*. The second one uses (11). From the identity

$$\sin^2(\varphi_1 + \varphi_2) - \sin^2(\varphi_1 - \varphi_2) = \sin 2\varphi_1 \sin 2\varphi_2,$$

we deduce $\lambda^*\mu^* \geq 1$. After some calculations, we obtain the third one and consequently (12). □

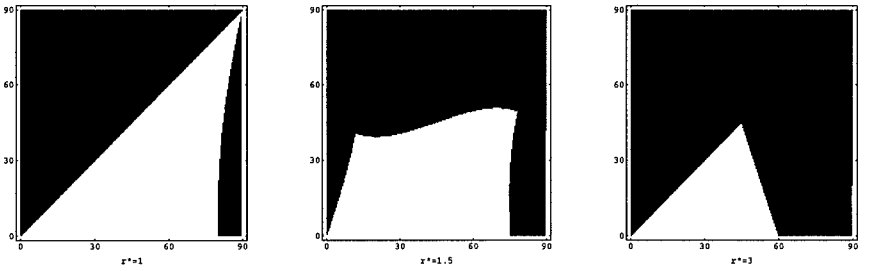


Fig. 1. Decreasing curvature domains for $r^* = 1, 1.5$ *and* 3.

Lemma 3. *Let P be a BCc curve defined on $[0,1]$. With the previous notation and hypothesis, we have*

$$r^* \cos(\varphi_1 + \varphi_2) - \cos 2\varphi_1 \geq 0 \Leftrightarrow \forall t \in [0,1], \quad r \geq 1.$$

Proof: For $t \in [0,1]$, we find that $(\Delta_1^{(0)}\Delta_1^{(1)})^2 - (\Delta_1^{(0)})^4$ is positive iff

$$(1-t)^2 A_0 + 2t(1-t)A_1 + t^2 A_2 \geq 0$$

with

$$\begin{cases} A_0 = 2(r^* \cos(\varphi_1 + \varphi_2) - \cos 2\varphi_1), \\ A_1 = 2h_1(r^* \cos\varphi_2 - \cos\varphi_1), \\ A_2 = h_1^2(r^{*2} - 1), \end{cases}$$

is positive. Then $r^* \cos(\varphi_1 + \varphi_2) - \cos 2\varphi_1 \geq 0$. Conversely, this inequality is equivalent to

$$(r^* \cos\varphi_2 - \cos\varphi_1)\cos\varphi_1 \geq (r^* \sin\varphi_2 - \sin\varphi_1)\sin\varphi_1.$$

Considering the cases $\varphi_1 \leq \varphi_2$ and $\varphi_2 \leq \varphi_1$, we see that the coefficients of the above Bernstein polynomial are positive. □

Proposition 1. *Let P be a BCc curve defined on $[0,1]$ with the representation $(h_1, h_2; \varphi_1, \varphi_2; l)$. Let us suppose that $r^* \geq 1$. We consider the domain*

$$D_1 = \{(\varphi_1, \varphi_2) \in [0, \frac{\pi}{2}), r^* \cos(\varphi_1 + \varphi_2) - \cos 2\varphi_1 \geq 0\}.$$

For $(\varphi_1, \varphi_2) \in D_1$, we have

$$\rho'(0) \leq 0 \Leftrightarrow \forall t \in [0,1], \quad \rho'(t) \leq 0.$$

Proof: The proof is a consequence of Lemmas 1 and 2. □

The domains corresponding to different values of r^* are described *via* the following graphs in function of φ_1 (horizontal axis) and φ_2 (vertical axis). In Figure 1, the white part represents the decreasing curvature domain whereas the dark one denotes a domain where the decrease at $t = 0$ is not possible ($r^*\lambda^* > 3$). One can notice that the latter is empty when $r^* > 6$. In the grey part, we cannot say anything about the monotonicity.

§5. Decreasing Curvature Condition in the Case $r^* < 1$

With $r^* < 1$, the condition $\rho'(0) \leq 0$ is too strict, so we consider the following sufficient condition on decreasing curvature at the origin:

$$\rho'(0) \leq 4\left(1 - \frac{1}{r^*}\right)\rho(0),$$

which is equivalent to

$$\frac{3}{2}\, h_1 \cos\varphi_1 \left(1 - \frac{r^*\lambda^*}{3}\right) \geq r^{*-1}. \tag{14}$$

If we set

$$\alpha^* = \frac{3}{2}\, h_1 \cos\varphi_1 \left(1 - \frac{r^*\lambda^*}{3}\right),$$

(14) becomes $r^*\alpha^* \geq 1$. Furthermore, we set $\forall t \in [0,1]$,

$$\alpha = \frac{3}{2}\, \Delta_1^{(0)} \cos\gamma_1^{(1)} \left(1 - \frac{r\lambda}{3}\right).$$

A calculation gives $\alpha = (1-t) + \alpha^* t$.

Remark 3 . The inequality (14) does not depend on the interval of definition.

Lemma 4. *Let P be a BCc curve defined on interval $[t_1, t_2]$ with the representation $(h_1, h_2; \varphi_1, \varphi_2; l)$ and $r^* < 1$. Let us consider domain*

$$D_2 = \{(\varphi_1, \varphi_2) \in [0, \frac{\pi}{2}[, r^* \cos\varphi_2 - \cos\varphi_1 \geq 0\}.$$

Then $\forall\ (\varphi_1, \varphi_2) \in D_2$,

$$\rho'(t_1) \leq 4\left(1 - \frac{1}{r^*}\right)\rho(t_1) \Rightarrow \rho'(t_2) \leq 0.$$

Proof: For $(\varphi_1, \varphi_2) \in D_2$, we have

$$\left(1 - \frac{r^*\lambda^*}{3}\right)\left(1 - \frac{r^*\mu^*}{3}\right)\cos\varphi_1 \cos\varphi_2 \leq \left(\frac{2}{3}\right)^2,$$

which, with (14), implies

$$h_2 = r^* h_1 \geq \frac{2}{3}\left(1 - \frac{r^*\lambda^*}{3}\right)^{-1} \frac{1}{\cos\varphi_1} \geq \frac{3}{2}\left(1 - \frac{r^*\mu^*}{3}\right)\cos\varphi_2. \qquad \square$$

Lemma 5. *Let P be a BCc curve defined on $[0,1]$. Under the previous hypothesis, we have*

$$\forall(\varphi_1,\varphi_2) \in D_2 \Leftrightarrow \forall t \in [0,1], \quad r\cos\gamma_1^{(2)} - \cos\gamma_1^{(1)} \geq 0.$$

Proof: For $t \in [0,1]$, we find that $r\cos\gamma_1^{(2)} - \cos\gamma_1^{(1)} \geq 0$ if and only if

$$(1-t)(r^*\cos(\varphi_1+\varphi_2) - \cos 2\varphi_1) + h_1(r^*\cos\varphi_2 - \cos\varphi_1)t \geq 0.$$

Then $r^*\cos\varphi_2 - \cos\varphi_1 \geq 0$. Conversely, $r^*\cos\varphi_2 - \cos\varphi_1 \geq 0$ and $r^* < 1$ imply $\varphi_2 \leq \varphi_1$. Then $r^*\sin\varphi_2 - \sin\varphi_1 \leq 0$. Consequently,

$$r^*\cos(\varphi_1+\varphi_2)-\cos 2\varphi_1=(r^*\cos(\varphi_1+\varphi_2)-\cos 2\varphi_1)\cos\varphi_1-(r^*\sin\varphi_2-\sin\varphi_1)\sin\varphi_1$$

is positive. □

Lemma 6. *Let P be a BCc curve defined on $[0,1]$. Under the previous hypothesis, we have $\forall(\varphi_1,\varphi_2) \in D_2$,*

$$h_1(r^{*2}\alpha^* - 1) + 2(r^*\alpha^{*2}\cos\varphi_2 - \cos\varphi_1) \geq 0 \Rightarrow \forall t \in [0,1], \quad r\alpha \geq 1.$$

Proof: The last inequality is equivalent to deciding whether a *fifth-degree* polynomial is positive. We verify that all its coefficients are positive but for one. The positivity of this coefficient is equivalent to the first inequality in the lemma. □

Proposition 2. *Let P be a BCc curve defined on $[0,1]$ with the representation $(h_1,h_2;\varphi_1,\varphi_2;l)$. Let us suppose that $r^* < 1$. We consider the domain D_2 as mentioned above and*

$$D_3 = \{(\varphi_1,\varphi_2) \in [0,\frac{\pi}{2}), h_1(r^{*2}\alpha^* - 1) + 2(r^*\alpha^{*2}\cos\varphi_2 - \cos\varphi_1) \geq 0\}.$$

Then, $\forall(\varphi_1,\varphi_2) \in D_2 \cap D_3$,

$$\rho'(0) \leq 4\left(1 - \frac{1}{r^*}\right)\rho(0) \Rightarrow \forall t \in [0,1], \quad \rho'(t) \leq 0.$$

Proof: The proof is a consequence of Lemmas 5,6 and 4. □

Here, we describe the different admissibility domains $D_2 \cap D_3$ (represented in white) for several values of r^* in considering that $\alpha^* = 1/r^*$. As illustrated in Figure 2, there is a domain continuity when r^* is near 1 with the case $r^* = 1$ (Figure 1). When r^* decreases, the domain gets smaller and smaller and then it is finally empty for $r^* \simeq 0,33$. If we take $\alpha^* \gg 1/r^*$, the domain size increases and tends to D_2.

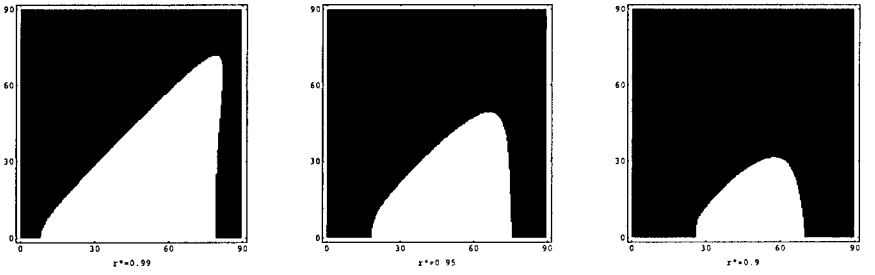


Fig. 2. Decreasing curvature domains for $r^* = 0.99, 0.95, 0.9$ *and* $\alpha^* = 1/r^*$.

§6. Examples

Our results are illustrated by examples of *BCc* curves whose curvature decreases. For each curve, one quarter of the curvature radius is represented at t values 0, 0.1, 0.2, ..., 1.

Example 1. $r^* = 1$ and $\rho'(0) = 0$. The first curve with small angles $(\frac{\pi}{7}, \frac{\pi}{16})$ and a length rate $h_1 \simeq 0.99$ (Figure 3 (a)) is characterized by a small radius increase. The second one has angles $(\frac{\pi}{4}, \frac{\pi}{8})$ and $h_1 \simeq 1.36$ (Figure 3 (b)).

Example 2. $(\varphi_1, \varphi_2) = (\frac{\pi}{3}, \frac{\pi}{4})$ and $\rho'(0) = 0$. The curve with $r^* = 1.25$ (Figure 4 (b)) is tighter than the curve with $r^* = 1$ (Figure 4 (a)).

Example 3. $r^* = 1.5$ and $(\varphi_1, \varphi_2) = (\frac{\pi}{5}, \frac{\pi}{4})$. We compare the curvature radius increase at the origin for a curve (Figure 5(a)) with $\rho'(0) = 0$ ($h_1 \simeq 1.71$) and another one (Figure 5 (b)) with $\rho'(0) = -0.12\ \rho(0)$ ($h_1 \simeq 2.05$).

Example 4. We consider two curves with $r^* < 1$ and $\alpha^* = 1/r^*$: $r^* = 0.95$, $(\varphi_1, \varphi_2) = (\frac{\pi}{4}, \frac{\pi}{5})$ and $r^* = 0.9$, $(\varphi_1, \varphi_2) = (\frac{\pi}{5}, \frac{\pi}{6})$. The parameters obtained by calculations (take the equality in (14)) are respectively $h_1 \simeq 1.44$, $\rho'(0) = -0.03\ \rho(0)$ and $h_1 \simeq 1.28$, $\rho'(0) = -0.06\ \rho(0)$ (Figure 6 (a)-(b)).

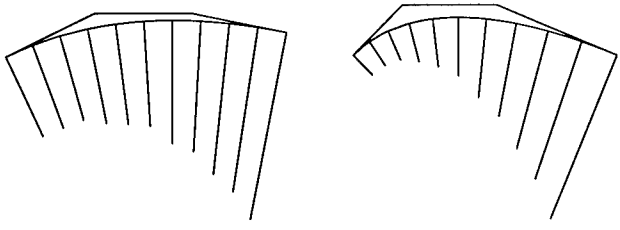


Fig. 3. (a) $r^* = 1$, $(\varphi_1, \varphi_2) = (\frac{\pi}{7}, \frac{\pi}{16})$, **(b)** $r^* = 1$, $(\varphi_1, \varphi_2) = (\frac{\pi}{4}, \frac{\pi}{8})$.

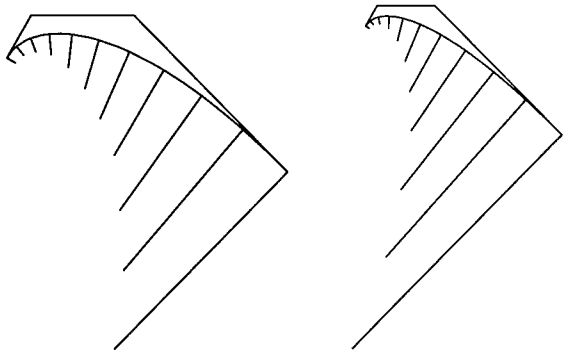


Fig. 4. **(a)** $r^* = 1$, $(\varphi_1, \varphi_2) = (\frac{\pi}{3}, \frac{\pi}{4})$, **(b)** $r^* = 1.25$, $(\varphi_1, \varphi_2) = (\frac{\pi}{3}, \frac{\pi}{4})$.

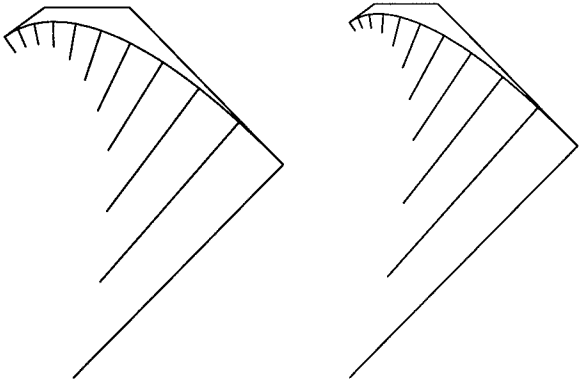


Fig. 5. $r^* = 1.5$ and $(\varphi_1, \varphi_2) = (\frac{\pi}{5}, \frac{\pi}{4})$ **(a)** $\rho'(0) = 0$, **(b)** $\rho'(0) = -0.12\rho(0)$.

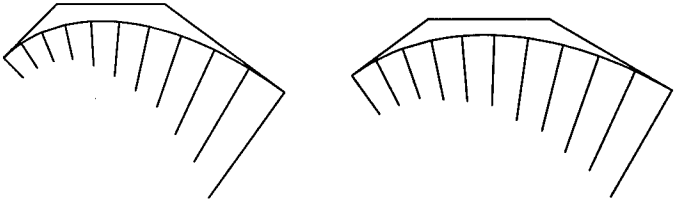


Fig. 6. **(a)** $r^* = 0.95$, $(\varphi_1, \varphi_2) = (\frac{\pi}{4}, \frac{\pi}{5})$, **(b)** $r^* = 0.9$, $(\varphi_1, \varphi_2) = (\frac{\pi}{5}, \frac{\pi}{6})$.

References

1. Fiorot, J.-C. and P. Jeannin, *Courbes et Surfaces Rationnelles. Applications à la CAO,* Masson, Paris, 1989. English version: *Rational Curves and Surfaces. Applications to CAD,* Wiley, Chichester, UK, 1992.
2. Fiorot, J.-C., P. Jeannin and S.Taleb, Rational cubic curves as BR-curves, Comput. Aided Geom. Design **12** (1995), 67-88.
3. Higashi, M., K. Kaneko and M. Hosaka, Generation of high quality curve and surface with smoothing varying curvature, in *Eurographics* '88, Elsevier Science, 1988, 79-92.
4. Mineur, Y., T. Lichah, J.-M. Castelain and H. Giaume, A shape controled fitting method for Bézier curves, Comput. Aided Geom. Design **15** (1998), 879-891.

Jean-Charles Fiorot
Université de Valenciennes et du Hainaut-Cambrésis
Laboratoire MACS, B.P.311
59304 Valenciennes Cedex, France
`fiorot@univ-valenciennes.fr`

Laurent Schiavon
Université de Valenciennes et du Hainaut-Cambrésis
Laboratoire MACS, B.P.311
59304 Valenciennes Cedex, France
`schiavon@univ-valenciennes.fr`

From Local Approximation to a G^1 Global Representation

Cédric Gérot, Dominique Attali, and Annick Montanvert

Abstract. To represent a complex surface, it is useful to describe it as a set of simple parametric primitives such as quadrics. But if one wants to use few primitives, these have to be smoothly blended. To define this blending, we propose to describe the initial global surface with *charts*. The blending surfaces result from a convex combination of primitives whose weights are defined on open sets of $\mathbb{R}^2$ given by the charts. We have established the properties that the weight functions must satisfy to obtain a G^1 representation of the global surface, and we have constructed such functions.

§1. Introduction

The abundance of high quality volumetric image data and new performant segmentation methods for multidimensional image data make 3-D objects ready for analysis. Volumetric objects are basically represented by a binary voxel representation or by a triangulation of the surface. Because they are based on huge lists of voxels or surface elements, they are not efficient for capturing global and local shape features with a view to characterizing shape properties. The spline surfaces can be very useful, but become difficult to use for topologically arbitrary surfaces modeling because they require a rectangular parameterization. On the contrary, any surface can be approximated using quadric surface patches as in [2]. While they lead to a good shape description, the quadric patches do not define an overall continuous surface. Ideally, a surface representation for image analysis should allow us to represent continuously any complex surface with few parameters, and to extract shape properties as well.

We propose to represent a surface with *charts*. A chart is composed of a patch U lying on the surface and a homeomorphism of U onto a 2D-domain. This notion has already been used, but essentially for image synthesis. Thus, in [14], it allows texture mapping on a triangulated surface which is too

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 109–118.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

complex to be described by only one chart. It is also used in [9] to design a surface with B-splines on any topological polyhedra, and in [15] to generalize the B-splines for constructing surfaces from irregular control meshes that can be embedded in the plane. Then, Eck et al. [5] use this notion to design a subdivision mesh from any triangular mesh.

We use it for image analysis because it allows to unfold a complex surface (for instance the surface of a brain). It is then an appropriate tool to extract surface features. Before using it, we first have to construct it. To do so, we begin by representing the surface by means of a set of simple surfaces called primitives (quadrics for instance) which approach it locally. (We currently study new processes to extract primitives from 3D objects). As mentioned above, the primitives do not define a globally continuous surface in general. So they have to be smoothly blended. This paper is focused on a solution to the blending problem.

Several different approaches to surface blending have been suggested. Firstly to fill a hole on a surface, one can interpolate a position and tangency conditions network [18,7,16,4], or construct a rational patch to fill a polygonal hole [10,8]. Our blending problem is not to fill a hole. But our approach solve this problem too. Secondly, to blend two surfaces, one can apply a rolling-ball algorithm [1,3,6]. But one cannot blend more than three surfaces at the same time. One can also meld isopotentials if the primitives are implicitly defined, [17,11,12]. But this seems to be a too restrictive condition. Our approach differs from these methods in that we blend any number of primitives at the same time, provided they can be parameterized.

This paper begins with mathematical definitions which are necessary to define our representation with charts. Then we present our approach for surface blending. Next we illustrate the different steps of our approach with some examples. Finally, we conclude with future work.

§2. Surface Representation with Charts

We begin with some mathematical definitions coming from differential geometry [13].

Definition 1. *A* n-dimensional manifold *is a topological space such that each point admits a neighborhood homeomorphic to* $\mathbb{R}^n$.

Definition 2. *A* chart (U, ψ) *is composed of an open set* U *of an n-manifold and a homeomorphism* ψ *of* U *onto an open set of* $\mathbb{R}^n$.

Definition 3. *Two charts* (U_i, ψ_i) *and* (U_j, ψ_j) *agree with each other if their* transition function

$$\psi_{ij} = \psi_j \circ \psi_i^{-1} \colon \psi_i(U_i \cap U_j) \to \psi_j(U_j \cap U_i)$$

is a diffeomorphism.

Definition 4. *Such a collection of maps charting all of the manifold is called an* atlas.

Remark. Every 2-dimensional manifold admits an atlas.

To give an atlas describing a surface is to give a representation of it with charts. This representation possesses two main advantages. Firstly, it combines local information with global information (ψ^{-1} is a local parameterization of the surface). That means one can work locally on the surface without undesirable consequences on the global surface because the atlas maintains this consistency by definition. Secondly, an atlas allows to translate problems given on any 2-dimensional manifold into problems given on $\mathbb{R}^2$.

§3. Smooth Blend

We have given the mathematical definition of the representation with charts. We now discuss how to use it to construct a G^1 global representation of a surface. Our aim is to represent a 2-dimensional manifold V by means of a set of simple surfaces called primitives (quadrics for instance) which approach V locally. To be more precise, we assume a family $\{U_i\}$ of open sets on V the union of which covers V and such that each U_i is approximated by a primitive P_i in such a way that there is a bijection b_i of U_i onto P_i: $P_i = b_i(U_i)$. The blend we want to construct between the primitives P_i must be a smooth surface S which overlaps a closed set of each P_i, called pure area and defined by $b_i(U_i \setminus \bigcup_{j \neq i} U_j)$. So, to be able to construct a smooth blend, the primitives must overlap sufficiently (see §4.1).

The blend is defined as a convex combination of the primitives P_i which approximate overlapping open sets U_i. The surface S is defined by an atlas and is a representation of V.

Hypotheses.

- We suppose that P_i are 2-dimensional manifolds parametrized by p_i, homeomorphisms which are C^1 on an open set Ω_i of $\mathbb{R}^2$: $P_i = p_i(\Omega_i)$.
- Let Ω_{ij} be the open set of Ω_i defined by $\Omega_{ij} = p_i^{-1}(b_i(U_i \cap U_j))$.
- We suppose that there exist some bijective transition functions $\varphi_{ij} : \Omega_{ij} \to \Omega_{ji}$, such that $\varphi_{kj} \circ \varphi_{ik} = \varphi_{ij}$. In particular, $\Omega_{ii} = \Omega_i$ and φ_{ii} is the identity. We write $\mathcal{P}(\mathbb{N})$ for the set of subsets of $\mathbb{N}$, and define for all i

$$\begin{aligned} \mathcal{I}_i : \Omega_i &\to \mathcal{P}(\mathbb{N}) \\ m &\mapsto \{j \in \mathbb{N} : \ m \in \Omega_{ij}\} \end{aligned}$$

- Let the weight functions α_i be defined on Ω_i and satisfying the following:

Property 1. Convexity

1a) $\forall i, \forall m \in \Omega_i, 0 \leq \alpha_i(m) \leq 1$,

1b) $\forall i, \forall m \in \Omega_i, \sum_{j \in \mathcal{I}_i(m)} \alpha_j(\varphi_{ij}(m)) = 1$.

Property 2. Regularity

2a) $\alpha_i(m) = 1$ if $p_i(m)$ belongs to the pure area,

2b) $\alpha_i(m) = 0$ if m does not belong to Ω_i.

- We define for all i

$$\varphi_i : \Omega_i \to \mathbb{R}^3$$
$$m \mapsto \sum_{j \in \mathcal{I}_i(m)} \alpha_j(\varphi_{ij}(m)) p_j(\varphi_{ij}(m)).$$

Remark. This definition is consistent: $\varphi_j(\varphi_{ij}(m)) = \varphi_i(m)$.

Proposition. With these hypotheses we get:

- If φ_i is bijective, φ_{ij} is C^0 and α_i is C^0, then S is a 2-dimensional manifold for which an atlas is $\{(\varphi_i^{-1}(\Omega_i), \varphi_i^{-1})\}$.
- If φ_{ij} is C^1 and α_i is C^1, then S is described by a C^1-atlas (S is then a G^1-surface).

Property 2a guarantees that $\varphi_i(m) = p_i(m)$ if $p_i(m)$ belongs to the pure area. Properties 2a and 2b can be inconsistent with each other if the pure area is not strictly included in $p_i(\Omega_i)$. But in this case, another primitive can be introduced, which overlaps locally P_i.

On one hand, our representation is more efficient if few primitives are used. On the other hand, S is closer to V when more primitives are used. So the appropriate balance must be found with regard to these needs. But, if the blend is not defined specifically to perform the approximation of V by S, the approximation error is on the same order of magnitude as that due to the local approximation by each primitive. This last property is due to the convexity property followed by the weight functions.

To construct weight functions which satisfy the convexity properties, we first construct functions β_i satisfying the following

Property 3.

3a) $\forall i, \forall m \in \Omega_i, 0 \leq \beta_i(m) \leq 1$,

3b) $\forall i, \forall m \in \Omega_i, \sum_{j \in \mathcal{I}_i(m)} \beta_j(\varphi_{ij}(m)) \neq 0$,

3c) $\beta_i(m) = 1$ if $p_i(m)$ belongs to the pure area,

3d) $\beta_i(m) = 0$ if m does not belong to Ω_i.

Then, the weight functions α_i defined by the following expression have all the desired properties:

$$\alpha_i(m) = \frac{\beta_i(m)}{\sum_{j \in \mathcal{I}_i(m)} \beta_j(\varphi_{ij}(m))},$$

where φ_{ij} is a C^1 transition function.

§4. Applications

In this paper, we detail the construction of the open sets Ω_i, the weight functions α_i, and the functions φ_i. Further work will focus on the construction of domains U_i and the transition functions φ_{ij}.

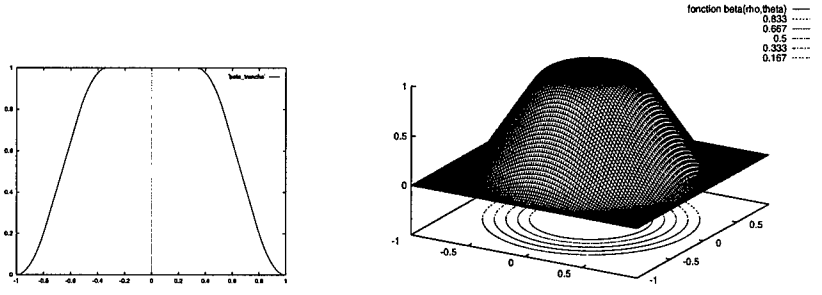


Fig. 1. Functions $b(\rho)$ and $\beta(\rho, \theta)$.

4.1. Weight functions

As shown in §3, to construct satisfactory weight functions, we first construct function β_i satisfying Property 3. We suppose the open set Ω is a disc whose radius is R. Let r be the radius of the smaller disc having the same center as Ω and including the set of points $\{m \in \Omega : p(m)$ belongs to the pure area$\}$. To simplify the notations, we call this set of points the pure area too. Then we can give a cylindric definition of β, where t is a shape parameter:

$$\beta(\rho, \theta) = b(\rho)$$

with

$$b(\rho) = \begin{cases} 1 & \text{if } \rho \leq r, \\ P(\rho) & \text{if } r < \rho \leq r + t, \\ L(\rho) & \text{if } r + t < \rho \leq R - t, \\ 1 - P(R + r - \rho) & \text{if } R - t < \rho \leq R, \\ 0 & \text{if } R < \rho, \end{cases}$$

where $L(x) = Dx + E$, $P(x) = Ax^2 + Bx + C$ and $D = \frac{-1}{R-r-t}$, $E = \frac{1-(R+r)D}{2}$, $A = \frac{D}{2t}$, $B = -2Ar$, $C = 1 - Ar^2 - Br$.

Fig. 2 shows the weight function α after normalization, in a case where P is combined with five other primitives.

In order that the small disc whose radius is r better fits the pure area, two modifications can be easily implemented. Firstly, we can define the small disc containing the pure area with different center than the center of Ω. β will have the same definition but with R depending on θ. Secondly, we can use ellipses rather than discs.

The parameter t, which belongs to $(0, 0.5)$, controls the nonlinear part of b. The smaller t, the smaller this part is. To avoid a final surface which is visually too sharp, t must be neither too small nor too large.

The size of the pure area also plays an important role in the surface smoothness. As shown in Fig. 4, if the pure area is too large, then the transitions between the primitives are too sharp in regards with the resolution of a visualization process. On the contrary, a pure area which is too small causes smooth transitions, but the shape of primitives is lost. In the example shown

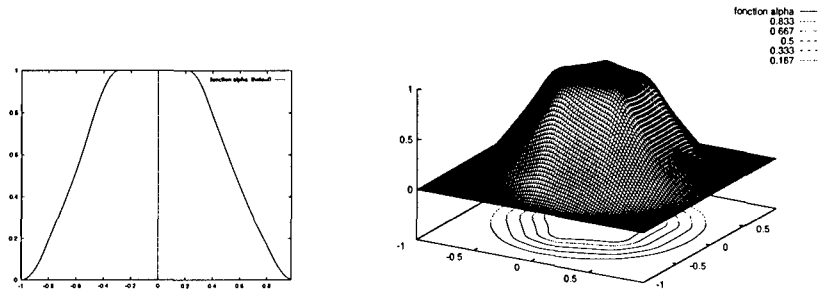


Fig. 2. Function α for a fixed θ and for all θ.

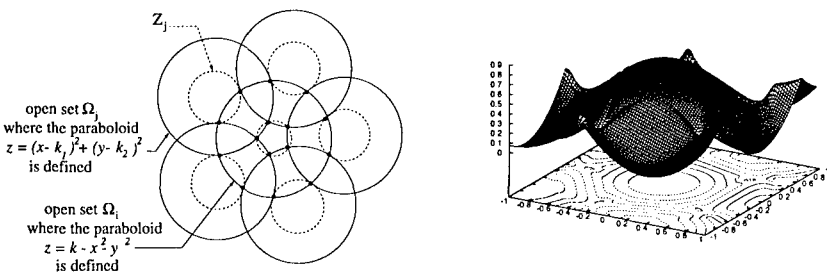


Fig. 3. Blending paraboloids: an example satisfying the hypotheses ($r = 0.71R$).

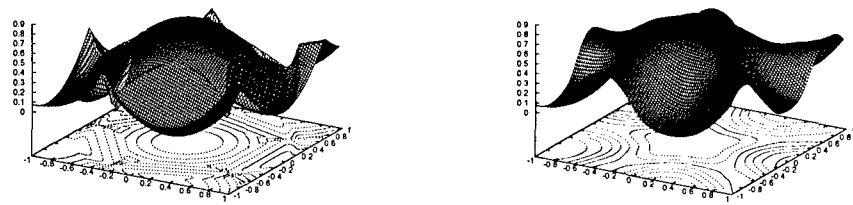


Fig. 4. Blendings with too large ($r = 0.85R$) and too small ($r = 0.07R$) pure areas.

in Fig. 3, we have implemented a case where the pure area is half the area of Ω. This balance gives a good solution.

We have constructed satisfactory weight functions. To apply our representation we must define the functions $\mathcal{I}_i$, bijective and C^1 transition functions φ_{ij}, and check that φ_i is bijective to be sure that S is a G^1 surface.

4.2. A simple case

We first apply our representation in the case where the surface V to be represented can be described by $v(x,y) = [x, y, f(x,y)]$, $(x,y) \in D \subset \mathbb{R}^2$: a land surface in topography for example.

We suppose that a set of open discs Ω_i is defined by any local approximation strategy, and parameterizations $p_i(x,y) = [x, y, p_i^z(x,y)]$, $(x,y) \in \Omega_i$ are given such that $D \subset \bigcup_i \Omega_i$ and each p_i is a C^1 approximation of $\{v(x,y) : (x,y) \in \Omega_i \cap D\}$.

It is easy to check if a point (x,y) is inside a disc Ω_j, and so to define $\mathcal{I}_i(x,y)$. Besides, the transition functions φ_{ij} are, in this case, the identity, which is C^1 and bijective. Finally, the functions φ_i constructed by convex combinations of such p_i are bijective. So, the surface S described by the atlas $\{(\varphi_i^{-1}(\Omega_i), \varphi_i^{-1})\}$ is G^1.

Fig. 3 shows an example of this first case. We deal with six open discs. One of them, Ω_i, is centered on the origin. The others are centered on the vertices of a pentagon which encircles Ω_i. The primitives are paraboloids. The central one is defined by $z = k - x^2 - y^2$, and the others by $z = (x - k_j^1)^2 + (y - k_j^2)^2$ where k_*^* are constant.

This example displays the blend between two primitives quite similar locally around their parts to blend (a central and a peripheral), and between two dissimilar primitives (two peripherals). The surface is smooth even if adjacent primitives are strongly different from each other.

4.3. A more general case

Most of the surfaces to be represented cannot be described by $[x, y, v^z(x,y)]$.

To deal with any surface V, we require a triangular mesh which is a first approximation of V. To simplify the notations, we name this mesh V too. We define on it a set of domains U_i. Each U_i is a set of vertices, edges and faces of V. It is isomorphic to an open disc, and well approximated by a primitive (a plane in Fig. 7).

In this case, we do not give an analytic expression to the functions p_i, φ_{ij} and the open set Ω_{ij}, but they are defined on a finite set of points. They are described by links between vertices of different meshes (see Fig. 5). For instance, we construct a mesh Ω_i lying on $\mathbb{R}^2$, using the bijective harmonic map presented in [5], on U_i, see Fig. 6. Therefore, each vertex u of U_i is linked to a vertex ω of Ω_i. Because of these links, $\mathcal{I}_i$, φ_{ij} and Ω_{ij} can be defined on the vertices of Ω_i as follows:

> For every vertex u of V, we construct $l(u)$, the list of the vertices linked to u. Each of these vertices lies on a different Ω_j. $l(u)$ contains only one vertex ω if ω belongs to a pure area. Let ω be a vertex of an open set Ω_i. Let u be the vertex of U_i (and so V) linked to it. For all i, if one of the vertices of $l(u)$, ω', belongs to Ω_j, then $\mathcal{I}_i(\omega)$ includes j, ω belongs to Ω_{ij} and $\varphi_{ij}(\omega) = \omega'$; or else ω does not belong to Ω_{ij}.

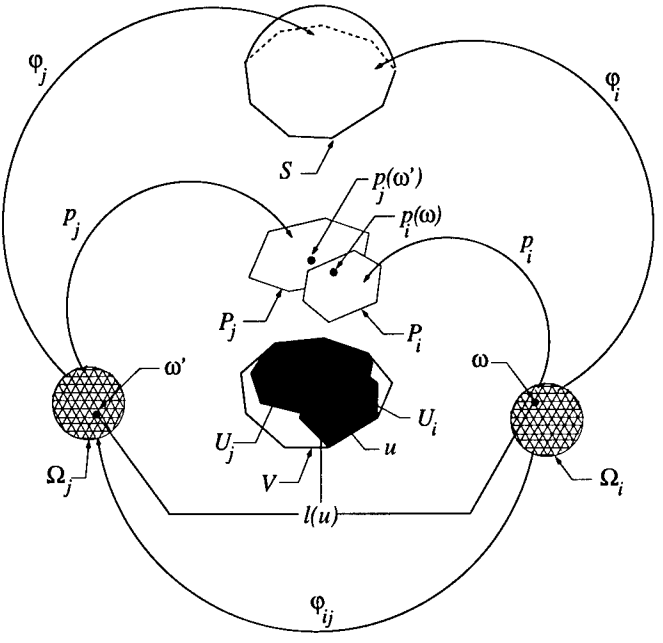


Fig. 5. Notation in the general case.

The functions β_i are calculated as in §4.1, and thanks to $\mathcal{I}_i$, we calculate α_i on the vertices of Ω_i.

In the same way, we construct a mesh P_i, which is in bijection with U_i and whose vertices lie on the primitive which approximate U_i. Thanks to the links between the vertices of U_i and Ω_i, we define the links between the vertices of Ω_i and P_i. These links define the paramatrization p_i on the vertices of Ω_i.

We then construct a mesh S whose vertices are calculated by φ_i defined on the vertices of Ω_i as in §3.

Assuming there exist C^1 functions φ_{ij} and p_i which interpolate the values set on the vertices of Ω_i, and satisfy the hypotheses given in §3, the vertices of S lie on a G^1 surface.

Remark. Because we do not give an analytic expression for p_i and φ_{ij}, we have to store the meshes Ω_i and P_i. In future work, we will either have to give simple expressions for those functions, or decrease the size of the meshes.

§5. Conclusion

The representation with charts can be used to construct a useful surface model. But, before applying it to real data, we still have two crucial steps: the definition of the domains U, and the definition of transition functions. Then we will apply it to image analysis problems such as registration, surface feature extraction, texture mapping or animation. More precisely, we will begin with the visualization of S by a mesh hierarchy which offers different levels of detail.

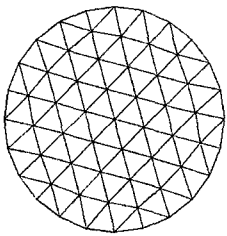


Fig. 6. An open set Ω_i.

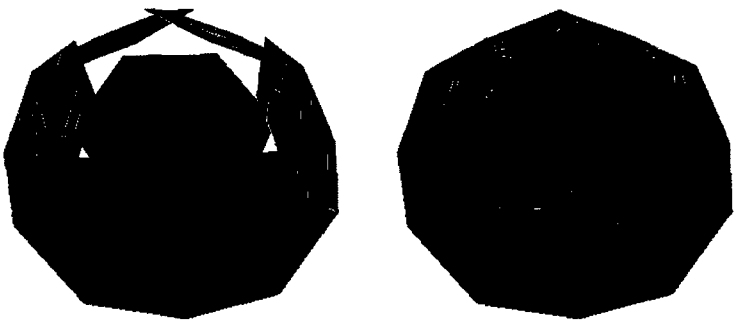


Fig. 7. The meshed primitives P_i and the mesh S.

References

1. Barnhill, R. E., G. E. Farin, and Q. Chien, Constant-radius blending of parametric surfaces, in *Geometric Modelling*, H. Hagen, G. Farin and H. Noltemeier (eds), Springer-Verlag, Austria, 1993, 1–20.
2. Bricault, I. and O. Monga, From volume medical images to quadratic surface patches, CVGIP:Image Understanding **67(1)** (1997), 24–38.
3. Chuang, J. H. and W. C. Hwang, Variable-radius blending by constrained spine generation, Visual Comput. **13** (1997), 316–329.
4. Costantini, P. and C. Manni, A bicubic shape-preserving blending scheme, Comput. Aided Geom. Design **13** (1996), 307–331.
5. Eck, M., T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuezle, Multiresolution analysis of arbitrary meshes, in *SIGGRAPH 95 Proc of Computer Graphics*, 1995, 173–182.
6. Filip, B. J., Blending parametric surfaces, ACM Trans. on Graphics **8(3)** (1989), 164–173.
7. Foley, T. A., S. Danayand, and R. Santhanam, Cross boundary derivatives for transfinite triangular patches, in *Geometric Modelling*, H. Hagen,

G. Farin and H. Noltemeier (eds), Springer-Verlag, Austria, 1993, 91–100.

8. Gregory, J. A., V. K. H. Lau, and J. M. Hahn, High order continuous polygonal patches, in *Geometric Modelling*, H. Hagen, G. Farin and H. Noltemeier (eds), Springer-Verlag, Austria, 1993, 117–132.
9. Grimm, C. M. and J. Hughes, Modeling surfaces of arbitrary topology using manifolds, in *SIGGRAPH 95, Proc. of Computer Graphics*, 1995, 359–367.
10. Hahn, J., Geometric continuous patch complexes, Comput. Aided Geom. Design **6** (1989), 55-67.
11. Hoffmann, C. and J. Hopcroft, Quadratic blending surfaces, Computer-Aided Design **18(6)** (1986), 301–306.
12. Hoffmann, C. and J. Hopcroft, The potential method for blending surfaces and corners, in *Geometric Modeling: Algorithms and New Trends*, G. E. Farin (ed), SIAM Publications, Philadelphia, 1987, 347–365.
13. Lehmann, D. and C. Sacré, *Géométrie et Topologie des Surfaces*, PUF,, France, 1982.
14. Maillot, J., H. Yahia, and A. Verroust, Interactive texture mapping, in *SIGGRAPH 93, Proc. of Computer Graphics*, 1993, 27–35.
15. Navau, J. C. and N. P. Garcia, Modelling surfaces from planar irregular meshes, Comput. Aided Geom. Design **17** (2000), 1–15.
16. Prautzsch, H., Algorithmic Blending, J. Approx. Theory **72** (1993), 87–102.
17. Rockwood, A. P. and J. C. Owen, Blending surfaces in solid modeling, in *Geometric Modeling: Algorithms and New Trends*, G. E. Farin (ed), SIAM Publications, Philadelphia, 1987, 367–383.
18. Varady, T., Overlap patches : a new scheme for interpolating curve networks with n-sided regions, Comput. Aided Geom. Design **8** (1991), 7–27.

Cédric Gérot, Dominique Attali and Annick Montanvert
Laboratoire LIS
ENSIEG
Domaine Universitaire
38402 St-Martin-d'Hères, France
`FirstName.LastName@lis.inpg.fr`

A Class of Totally Positive Blending B-Bases

Laura Gori, Laura Pezza, and Francesca Pitolli

Abstract. Totally positive blending bases present good shape preserving properties when they are used in CAGD. Among these bases there exist special bases, called B-bases, which have optimal shape preserving properties. In particular, the corresponding control polygon is nearest to the curve among all the control polygons; thus many geometrical properties are similar to the ones of the curve. Examples of totally positive blending B-bases are the Bernstein polynomials and the B-spline basis. Our purpose is to construct new classes of such bases starting from compactly supported totally positive scaling functions.

§1. Introduction

One of the main goals in Computer Aided Geometric Design (CAGD) is to predict or control the *shape* of a curve by studying or specifying the *shape* of the control polygonal arc formed by certain points which define the curve, typically the coefficients when the curve is expressed in terms of a particular basis. This is possible when we choose as a basis a system of functions $\boldsymbol{v} = (v_0, \dots, v_n)$ with suitable shape preserving properties. This means that the geometrical properties of the curve in $\mathbb{R}^2$

$$\gamma(x) = \sum_{i=0}^{n} P_i v_i(x), \quad x \in I \subset \mathbb{R}, \tag{1}$$

constructed on the control points $P_i \in \mathbb{R}^2$, $i = 0, \dots, n$, are implied by the geometrical properties of the control polygon $P_0 \dots P_n$. The shape preserving properties of each representation (1) depend on the characteristic of the system $\boldsymbol{v}$.

The bases commonly used in CAGD, such as Bernstein bases, B-splines, β-splines, nonuniform rational splines (NURBS), are blending totally positive systems. This means that the collocation matrix

$$M \begin{pmatrix} v_0, \dots, v_n \\ x_0, \dots, x_s \end{pmatrix} := (v_i(x_j))_{i=0\ j=0}^{n\ \ s} \tag{2}$$

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 119–126.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

for any sequence $x_0 < \cdots < x_s$, $x_i \in I$, $i = 0, \ldots, s$, is totally positive (*i.e.*, all its minors are non-negative), and the basis functions add to one, that is

$$\sum_{i=0}^{n} v_i(x) = 1, \quad x \in I. \tag{3}$$

The importance of blending totally positive systems is due to the fact that they enjoy two properties which are usually demanded for curve control: the convex hull (CH) and the variation diminishing (VD) properties (see, for instance, [5,6]). As a consequence, in many ways the shape of the curve γ mimics the shape of the control polygon $P_0 \ldots P_n$. However, blending totally systems usually do not enjoy a property which is also important: the end-point interpolation (EPI) property.

Bases which simultaneously satisfy the VD, CH and EPI properties can be obtained by considering blending B-bases [5].

Following [5], a totally positive (TP) system $\boldsymbol{u}$ of linearly independent functions is said to be a B-basis if any totally positive basis $\boldsymbol{v}$ of the space U generated by $\boldsymbol{u}$ satisfies the condition

$$\boldsymbol{v} = \boldsymbol{u}A, \quad A \text{ nonsingular totally positive matrix.} \tag{4}$$

In [4] it was proved that if there exists a blending TP basis in U, then there exists a *unique* blending B-basis for that space. B-bases have *optimal properties* in the geometric context [5], that is, in particular, the control polygon with respect to the B-basis is nearest to the curve among all the control polygons with respect to any other TP basis.

Some examples of B-bases are given in [4,5]; in particular, the B-spline basis is the blending B-basis in the space of the polynomial splines of degree m, on a given interval with a prescribed sequence of knots.

At this point, it is worthwhile to remark that in the case of cardinal splines (knots at the integers), this basis is related to the cardinal B-spline N^m, defined by $N^m = N^{m-1} * N^0$, where N^0 is the characteristic function of $[0, 1)$ and $*$ denotes the convolution product (see, for instance [8]).

On the other hand, N^m is a scaling function, that is the solution of the functional equation

$$N^m(x) = \frac{1}{2^m} \sum_{i=0}^{m+1} \binom{m+1}{i} N^m(2x - i), \quad x \in \mathbb{R}. \tag{5}$$

In this paper, we analyse the more general problem of the construction of blending B-bases considering, instead of N^m, a scaling function satisfying a functional equation more general than (5):

$$\varphi(x) = \sum_{i \in \mathbb{Z}} a_i \varphi(2x - i), \quad x \in \mathbb{R}, \tag{6}$$

where the mask $\boldsymbol{a} = \{a_i\}_{i\in\mathbb{Z}}$ satisfies the following conditions:

$$\sum_{i\in\mathbb{Z}} a_{2i+1} = \sum_{i\in\mathbb{Z}} a_{2i} = 1. \tag{7}$$

It is known that a solution φ of (6) exists if the mask $\boldsymbol{a}$ satisfies further conditions, in addition to (7). In particular, if:

i) $\boldsymbol{a}$ is compactly supported on $[0, m+1]$ (with $a_0 a_{m+1} \neq 0$),

ii) the symbol

$$p(z) = \sum_{i=0}^{m+1} a_i z^i \tag{8}$$

has roots with negative real part (Hurwitz polynomial),

then there exists [8] a *unique* scaling function solution of (6), whose support is $[0, m+1]$, such that

$$\sum_{i\in\mathbb{Z}} \varphi(x-i) = 1, \quad x \in \mathbb{R}. \tag{9}$$

Moreover, the functions $\{\varphi(\cdot - i), i \in \mathbb{Z}\}$ are linearly independent and totally positive on $\mathbb{R}$.

The aim of this paper is to construct new classes of blending B-bases, from a given system $\{\varphi(\cdot - i), i \in Z\}$, where Z is a finite subset of $\mathbb{Z}$ and φ is a scaling function. In Section 2 some preliminaries are outlined, whereas in Section 3 this construction is specialized to the new classes of scaling functions introduced in [10]. Finally, Section 4 is devoted to some examples.

§2. Preliminaries

Let $I = [\alpha, \beta]$, with α, β integers, be a finite interval of $\mathbb{R}$ and let φ be a compactly supported scaling function, whose support is $[0, L]$, associated with a mask $\boldsymbol{a}$ enjoying the properties *i)* and *ii)* of the previous section. Then, the system of $n = \beta - \alpha + L - 2$ functions

$$\boldsymbol{\Phi} := \{\varphi(x-i), \alpha - L + 1 \leq i \leq \beta - 1\}, \quad x \in [\alpha, \beta], \tag{10}$$

constitutes a blending (cf. (9)) TP basis in the space U_Φ generated by itself, and fulfils some interesting shape preserving properties.

Indeed, because of the properties of φ mentioned above, the basis $\boldsymbol{\Phi}$ satisfies the CH and the VD properties. Thus, $\boldsymbol{\Phi}$ preserves *monotonicity* and *convexity*, that is, any straight line cuts the curve γ_Φ no more often than it cuts the control polygon [7]. Further shape preserving properties can be deduced by the generalized VD property for TP bases (see [2]).

It is rather natural to wonder whether $\boldsymbol{\Phi}$ is a B-basis, too. To this end we can use the following proposition from [4].

Proposition A. *A TP basis $B = (\zeta_0, \dots, \zeta_n)$ is a B-basis if and only if the following conditions hold:*

$$\inf\left\{\frac{\zeta_i(x)}{\zeta_j(x)} \mid x \in I, \zeta_j(x) \neq 0\right\} = 0,$$

for all $i \neq j$.

Clearly, Proposition A provides a useful test to check if a TP basis is a B-basis. If the check fails, one can construct the unique blending B-basis of the space U_Φ by means of the procedure given in [4, Th 3.6 and Th. 4.2].

§3. Construction of B-bases of Scaling Functions

One of the main advantages of the cardinal B-spline as scaling function is that its mask has an explicit expression (cf. (5)). A wide generalization of the cardinal B-splines was developed in [10], where a new family of scaling functions has been introduced by means of a *new family of masks*, which have an explicit expression. These scaling functions depend on certain free parameters, have prescribed smoothness and, as for the cardinal B-splines, are compactly supported, totally positive and centrally symmetric. They were introduced as follows.

Let $\boldsymbol{H}$ denote the set of all compactly supported and centrally symmetric masks whose symbol is a Hurwitz polynomial. In [10] it was proved that a mask $\boldsymbol{a}$ belongs to $\boldsymbol{H}$ if and only if its coefficients are of the type

$$a_i^{(m,k)} = \sum_{r=0}^{k/2} b_r^{(r)} \binom{m+1-2r}{i-r}, \qquad i = 0, 1, \dots, m+1, \tag{11}$$

where $m = 2, 3, \dots$, k is an even integer such that $1 \leq k \leq m$, and

$$b_i^{(r)} = b_i^{(r-1)} - \binom{k-2r+2}{i-r+1} b_{r-1}^{(r-1)}, \quad \begin{array}{l} r = 0, 1, \dots, K, \quad K := \dfrac{k}{2} - 1, \\ i = r+1, \dots, K+1, \end{array} \tag{12}$$

and $b_i^{(0)}$, $i = 0, \dots, k$, are such that

$$\begin{cases} b_{k-r}^{(0)} = b_r^{(0)}, \quad r = 0, 1, \dots, k, \\ b_{\frac{k}{2}}^{(0)} = 2^{k-m} - 2\displaystyle\sum_{i=0}^{K} b_i^{(0)}, \\ \det\left(b_{2i-j}^{(0)}; i, j = 1, \dots, p\right) > 0, \quad p = 1, \dots, k \end{cases} \tag{13}$$

(assume $\binom{l}{i} = 0$ for $i < 0$ or $i > l$).

Due to the properties of $\boldsymbol{a} \in \boldsymbol{H}$, the scaling function $\varphi_{m,k}$, which is the solution of the scaling equation

$$\varphi_{m,k}(x) = \sum_{i=0}^{m+1} a_i^{(m,k)} \varphi_{m,k}(2x - i), \quad x \in \mathbb{R}, \tag{14}$$

is compactly supported on $[0, m+1]$ and centrally symmetric, and the functions $\{\varphi_{m,k}(\cdot - i), i \in \mathbb{Z}\}$ are linearly independent, normalized and TP. Moreover, recalling that a scaling function belongs to $C^r(\mathbb{R})$ if and only if the symbol can be factored as

$$p(z) = (z+1)^{r+1} q_{m-r}(z), \quad q_{m-r}(1) = 2^{-r}, \tag{15}$$

(see [8]), one can prove that $\varphi_{m,k} \in C^{m-k}(\mathbb{R})$.

Remark. *Choosing suitably the coefficients $b_i^{(0)}$, the $\varphi_{m,k}$ reduces to the cardinal B-spline of degree m, and the $\varphi_{m,k}$ can be viewed as a generalization of the cardinal B-splines. In particular, for $k = 1$, the unique family of scaling functions that we obtain are the cardinal B-splines. Moreover, in the case when $m = 3$, the coefficients of the mask (11) are a subset of those of the filters exploited by Burt and Adelson in vision analysis* [1].

Following the procedure outlined in the previous section, any of the scaling functions $\varphi_{m,k}$ can be used to construct a blending TP basis $\boldsymbol{\Phi}_{m,k}$ defined on a finite interval. Observe that a space is suitable for design purposes if it has a blending TP basis.

By means of Proposition A, it is easy to show that the basis $\boldsymbol{\Phi}_{m,k}$ is not a B-basis. Then to obtain a blending B-basis starting from the functions $\varphi_{m,k}(x - i)$, we have to apply the procedure given in [4]. The corresponding algorithm can be illustrated as follows. Let

$$u_i^0 = \varphi_{m,k}(x - i), \quad i = \alpha - m, \ldots, \beta - 1,$$

where the values of $\varphi_{m,k}$ can be evaluated by means of the cascade algorithm [12]. For $j = 0, \ldots, m-2$, define iteratively

$$u_i^{j+1} := \begin{cases} u_i^j - \inf\left(\dfrac{u_i^j}{u_{i-1}^j}\right) u_{i-1}^j, & i = m, m-1, \ldots, j+1, \\ u_i^j, & i = j, j-1, \ldots, 0. \end{cases}$$

Then, let

$$v_i^0 = u_i^{m-1}, \qquad i = \alpha - m, \ldots, \beta - 1,$$

and for $j = 0, \ldots, m-2$ define iteratively

$$v_i^{j+1} := \begin{cases} v_i^j - \inf\left(\dfrac{v_i^j}{v_{i+1}^j}\right) v_{i+1}^j, & i = 0, 1, \ldots, \beta - 2 - j, \\ v_i^j, & i = \beta - 1 - j, \ldots, \beta - 1. \end{cases}$$

The system $\Phi^B_{m,k} = \{b_i := v_i^{m-1}, i = \alpha - m, \dots, \beta - 1\}$, forms a B-basis. The system $\{d_i b_i, i = \alpha - m, \dots, \beta - 1\}$, where d_i, $i = \alpha - m, \dots, \beta - 1$ are positive constants such that $d_{\alpha-m} b_{\alpha-m} + \cdots + d_{\beta-1} b_{\beta-1} = 1$, is the required blending B-basis.

We remark that one of the difficulties in applying this method lies in the evaluation of $\inf(u_i^j / u_{i-1}^j)$ and $\inf(v_i^j / v_{i+1}^j)$. For instance, in the examples of Section 4, the infimums has been evaluated by extrapolating the values that the involved functions u_i and v_i assume in a suitable right neighbourhood of α and in a suitable left neighbourhood of β, respectively.

§4. Examples

For $k = 2$, the mask (11) depends on a free parameter $b_0^{(0)}$, which for computational convenience we chose as a dyadic fraction: $b_0^{(0)} = 2^{-h}$. Thus, the explicit expression of the mask coefficients becomes

$$a_{j,m}^{(h)} = 2^{-h} \left[\binom{m+1}{j} + 4(2^{h-m} - 1) \binom{m-1}{j-1} \right], \tag{16}$$

$(j = 0, 1, \dots, m+1,\ m \geq 2,\ h > m - 1)$, which corresponds to the symbol

$$p_{m,h}(z) = 2^{-h}(1+z)^{m-1}(z^2 + (2^{h-m+2} - 2)z + 1). \tag{17}$$

Observe that the second term in the mask (16) can be seen as a perturbation of the mask of the cardinal B-spline to which (16) reduces when $h = m$.

Given the interval $I = [\alpha, \beta]$, we can construct the family of blending TP bases

$$\mathbf{\Phi}_{m,h} = \{\varphi_{m,h}(x - i), \alpha - m \leq i \leq \beta - 1\}, \tag{18}$$

where $m \geq 2$ and $h > m - 1$. In Fig. 1 the basis $\mathbf{\Phi}_{3,4}$ defined on the interval $[0, 4]$ is displayed (dashed line) together with the corresponding blending B-basis (solid line) obtained by means of the procedure outlined in the previous section.

For $k = 4$, the symbol $p(z)$ depends on two free parameters, that is $b_0^{(0)}$ and $b_1^{(0)}$, which again, for computational convenience, we choose as dyadic fractions: $b_0^{(0)} = 2^{-h}$, $b_1^{(0)} = 2^{l-h}$; $h, l \in \mathbb{R}$ are arbitrary numbers such that $h > m - 2 + \log_2(1 + 2^{l-1})$, in order to fulfil the third of (13). Thus, the symbol has the form

$$p_{m,h,l}(z) = 2^{-h}(1+z)^{m-3} \left(z^4 + 2^l z^3 + (2^{-m+4+h} - 2 - 2^{l+1}) z^2 + 2^l z + 1 \right), \tag{19}$$

where $m > 3$, and the coefficients $a_{i,m}^{(h,l)}$, $0 \leq i \leq m+1$, of the corresponding mask are

$$a_{i,m}^{(h,l)} = \frac{1}{2^h} \left[\binom{m+1}{i} + (2^l - 4) \binom{m-1}{i-1} + (2^{-m+4+h} - 2^{l+2}) \binom{m-3}{i-2} \right]. \tag{20}$$

Also in this case, the mask of the cardinal B-spline N^m can be obtained setting suitably the parameters h and l, that is, $h = m$ and $l = 2$. In Fig. 2 the blending TP basis $\mathbf{\Phi}_{5,6,2}$ defined on the interval $[0, 6]$ is displayed (dashed line) together with the corresponding blending B-basis (solid line) obtained.

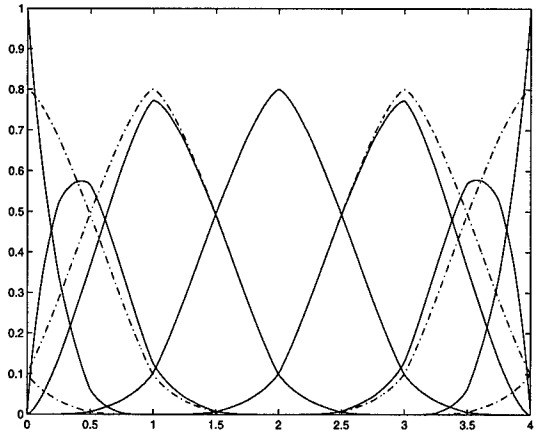


Fig. 1. The blending B-basis $\Phi^B_{3,4}$ (solid line) and the blending TP basis $\Phi_{3,4}$ (dashed line) in the interval $[0, 4]$.

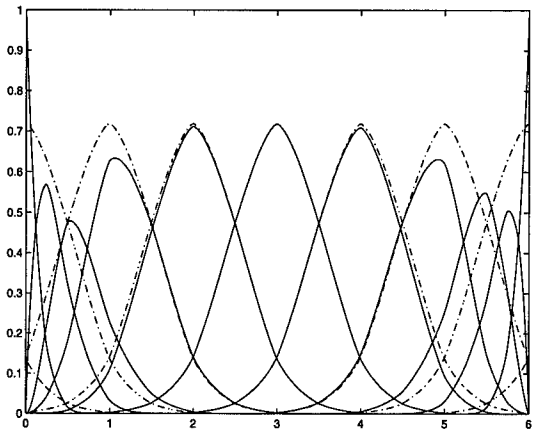


Fig. 2. The blending B-basis $\Phi^B_{5,6,2}$ (solid line) and blending TP basis $\Phi_{5,6,2}$ (dashed line) in the interval $[0, 6]$.

Remark. *When the scaling function is just N^m, the procedure outlined here gives the basis of the cardinal B-splines as defined in* [11].

Acknowledgments. This research was partially supported by the Italian Ministry of University and Scientific and Technological Research.

References

1. Burt, P., and E. Adelson, The laplacian pyramid as a compact image code, IEEE Trans. Comm. **31** (1983), 482–540.
2. Carnicer, J. M., T. N. T. Goodman and J. M. Peña, A generalization of the variation diminishing property, Advances in Comp. Math. **3** (1995), 375–394.
3. Carnicer, J. M., and J. M. Peña, Shape preserving representation and optimality of the Bernstein basis, Advances in Comp. Math. **1** (1993), 173–196.
4. Carnicer, J. M., and J. M. Peña, Totally positive bases for shape preserving curves design and optimality of B-splines, Comput. Aided Geom. Design **11** (1994), 633–654.
5. Carnicer, J. M., and J. M. Peña, Total positivity and optimal bases, in *Total Positivity and its Applications*, M. Gasca and C. A. Micchelli (eds.), Kluwer Academic Publishers, 1996, 133–155.
6. Farin, G., *Curves and surfaces for computer aided geometric design: a practical guide*, Academic Press, Inc., San Diego, 1988.
7. Goodman, T. N. T., Shape preserving representations, in *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker (eds.), Academic Press, New York, 1989, 333–357.
8. Goodman, T. N. T., and C. A. Micchelli, On refinement equations determined by Pólya frequency sequence, SIAM J. Math. Anal. **23** (1992), 766–784.
9. Goodman, T. N. T., and H. B. Said, Shape preserving properties of the generalized Ball basis, Comput. Aided Geom. Design **8** (1991), 115–121.
10. Gori, L., and F. Pitolli, A class of totally positive refinable functions, submitted.
11. Schumaker, L. L., *Spline functions: Basic Theory*, Krieger Publishing Company, Malabar, Florida, 1993.
12. G. Strang, T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley MA, 1996.

Laura Gori, Laura Pezza, and Francesca Pitolli
Dept. MeMoMat
Università *La Sapienza*
Via Antonio Scarpa 16 - 00161 Roma, Italy
gori@dmmm.uniroma1.it
pezza@dmmm.uniroma1.it
pitolli@dmmm.uniroma1.it

Rational Ruled Surfaces Passing Through Two Fixed Lines

Gueorgui H. Gueorguiev

Abstract. For any positive integer n, a rational ruled surface of degree $n+1$ is constructed which passes through two arbitrary skew lines in the three-dimensional Euclidean space. In the cases of two parallel or intersecting lines, a rational ruled surface of degree $2n+1$ is constucted which contains the lines. Any surface is a preimage of a plane under a birational space transformation. This interpretation gives implicit equations and parametric representations of the considered surfaces.

§1. Introduction

Ruled surfaces play an important role in computer aided geometric design (see [2,5,6]). In this paper, we construct rational ruled surfaces which are generalizations of the hyperbolic paraboloid and the hyperboloid of one sheet. Our main results describe three families of ruled surfaces which pass through two skew lines, two parallel lines, and two intersecting lines. The resulting surfaces in three-dimensional Euclidean space $\mathbb{R}^3$ (especially their parts without singularities) can be used in engineering and manufacturing. These ruled surfaces are found by the use of some birational transformations of $\mathbb{R}^3$. This interpretation also provides a way for finding an implicit equation and parametric representation of any such surface.

The paper is organized as follows. We introduce special birational transformations of the projective space $\mathbb{P}^3$ in Section 2. Any transformation determines a three-parameter family of surfaces whose images are planes. Corresponding transformations and rational surfaces in $\mathbb{R}^3$ are described in Section 3. For any positive integer n, a rational ruled surface of degree $n+1$ is constructed which passes through two arbitrary skew lines in the next section. The rational ruled surfaces of odd degree passing through two parallel or intersecting lines are considered in the last two sections.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 127–134.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

§2. Birational Transformations of the Projective 3-Space

Any rational surface is birational equivalent to a plane. In particular, the preimage of a plane under a birational space transformation is a rational surface. In this section, we shall consider birational transformations such that the preimage of an arbitrary plane is a rational ruled surface. First, we briefly recall some basic notions for birational transformations.

Definition 1. *Let $\mathbb{P}^3$ be the three-dimensional complex projective space. The map of $\mathbb{P}^3$ into itself*

$$T : \mathbb{P}^3 \to \mathbb{P}^3$$

is called a birational transformation *if there exists an open subset $U \subset \mathbb{P}^3$ in the Zariski topology such that the restriction $T|_U : U \to U$ is a one-to-one correspondence.*

In terms of homogeneous coordinates, the map T is birational, if

i) T is given by the equations

$$\rho' X_i' = F_i'(X_0, X_1, X_2, X_3), \qquad i = 0, 1, 2, 3, \tag{1}$$

where F_i' are homogeneous polynomials of the same degree and ρ' is a nonzero factor of proportionality;

ii) The inverse map T^{-1} exists and is given by the equations

$$\rho'' X_i'' = F_i''(X_0, X_1, X_2, X_3), \qquad i = 0, 1, 2, 3, \tag{2}$$

where F_i'' are also homogeneous polynomials of the same degree and $\rho'' \neq 0$.

In (1) and (2), the quadruples (X_0, X_1, X_2, X_3), (X_0', X_1', X_2', X_3') and $(X_0'', X_1'', X_2'', X_3'')$ are homogeneous coordinates of the points $q \in \mathbb{P}^3$, $q' = T(q)$ and $q'' = T^{-1}(q)$, respectively. Moreover, it is possible that $\deg F_i' \neq \deg F_i''$.

A fundamental (or base) locus of the birational transformation T given by (1) is the variety of common zeros of the polynomials F_i'. There is a three-parameter family of rational surfaces such that the image of any surface under T is a plane. Then, the intersection of all such surfaces is the fundamental locus of T. Note that the birational transformations of the projective space are also called Cremona transformations. More information for the birational transformarions can be found in [1] and [4].

Now, we shall study a class of birational space transformations. For any three fixed numbers $\theta_1 \in \mathbf{C} \setminus \{0\}$, $\theta_2 \in \mathbf{C} \setminus \{0\}$ and $\theta \in \mathbf{C} \setminus \{0, 1\}$, and for any positive integer n, we may consider the map $T_0 : \mathbb{P}^3 \to \mathbb{P}^3$ given by the equations

$$\begin{aligned}
\rho X_0' &= (\theta_1 X_0^n - \theta_2 X_3^n) X_0, \\
\rho X_1' &= (\theta_1 X_0^n - \theta_2 X_3^n) X_1 - (1-\theta)\theta_1 (X_1 X_0^n - X_2 X_3^n), \\
\rho X_2' &= (\theta_1 X_0^n - \theta_2 X_3^n) X_2 - (1-\theta)\theta_2 (X_1 X_0^n - X_2 X_3^n), \\
\rho X_3' &= (\theta_1 X_0^n - \theta_2 X_3^n) X_3.
\end{aligned} \tag{3}$$

From the condition $\theta \neq 1$, it follows that T_0 is not the identity mapping.

Theorem 1. *The map T_0 is a birational transformation, and the reducible curve*

$$B : \begin{cases} \theta_1 X_0^n - \theta_2 X_3^n & = 0 \\ X_1 X_0^n - X_2 X_3^n & = 0 \end{cases}$$

is the fundamental locus of T_0.

Proof: Let C be the surface given by the equation $\theta_1 X_0^n - \theta_2 X_3^n = 0$. Then, the map T_0 is a one-to-one correspondence in the set $\mathbb{P}^3 \setminus C$ which is open in the Zariski topology. On the other hand, the inverse mapping T_0^{-1} is defined by the equations (3) in which θ is replaced by θ^{-1}. Thus, T_0 is birational. The curve B is the set in which T_0 is not defined. Hence, B is the fundamental locus of T_0. □

The linear transformations of $\mathbb{P}^n$ $(n = 1, 2, 3)$ and their invariant, a cross-ratio, are studied in detail in [7]. Some geometric properties of the nonlinear transformation T_0 can be described in terms of a cross-ratio and collineations. The line $S_{03} \subset \mathbb{P}^3$ given by $X_0 = X_3 = 0$ is the $n-$fold line of the ruled surface G given by the equation $X_1 X_0^n - X_2 X_3^n = 0$. Let c be the point with homogeneous coordinates $(0, \theta_1, \theta_2, 0)$. Then, for any point $q \in \mathbb{P}^3 \setminus C$, the joining line $\overline{cq}$ meets G at the second point $q^0 \neq c$. From here, the point $q' = T_0(q)$ lies on the line $\overline{cq}$, and the cross-ratio $\{c, q^0; q, q'\} = \theta$. Continuing in this way, we consider a plane P given by $\lambda_0 X_0 + \lambda_3 X_3 = 0$, where $(\lambda_0, \lambda_3) \in \mathbf{C}^2 \setminus \{(0,0)\}$ and $\lambda_0^n : \lambda_3^n \neq \theta_1 : \theta_2$. Then, the intersection $P \cap G$ falls into $n-$fold line S_{03} and another line H not containing the point c. From the equations (3), we may conclude that T_0 preserves P and the restriction $T_0|_P : P \to P$ is a plane homology with a vertex c, an axis H and a modulus θ. It is clear that the set of all fixed points of T_0 is $G \setminus B$.

Theorem 2. *Let V be a surface in $\mathbb{P}^3$ such that the image $T_0(V)$ is a plane. Then, V is a rational ruled surface of degree $n+1$. In the case $n \geq 2$, the singular locus of V is an $n-$fold line S_{03}.*

Proof: From the (3) it follows that the surface V is given as the locus of

$$(\theta_1 X_0^n - \theta_2 X_3^n)(\sum_{i=0}^{3} \lambda_i X_i) - (1 - \theta)(\lambda_1 \theta_1 + \lambda_2 \theta_2)(X_1 X_0^n - X_2 X_3^n) = 0, \quad (4)$$

where $\lambda_i \in \mathbf{C}$ for $i = 0, 1, 2, 3$ and $\sum_{i=0}^{3} |\lambda_i| \neq 0$. Hence, $\deg V = n + 1$. By (4), if P is a plane through the line S_{03}, then the intersection $P \cap V$ falls into the $n-$fold line S_{03} and another line L. This means that V is a ruled surface. It is known from [3] that the singular locus of a ruled surface is connected. Thus, $\mathrm{Sing}(V) = S_{03}$. □

§3. Birational Transformations of the Euclidean 3-Space

Suppose that θ_1, θ_2 and θ are nonzero real numbers. Then, by (3), this defines a birational transformation of the 3-dimensional real projective space, or equivalently, a birational transformation of the projective extension of the Euclidean 3-space. Thus, we get a birational transformation of the Euclidean 3-space.

Theorem 3. *Let n be a positive integer, θ_1 and θ_2 be positive real numbers, and let $\theta \in \mathbb{R} \setminus \{0,1\}$. Then, the transformation $T_1 : \mathbb{R}^3 \to \mathbb{R}^3$ given by the equations in Cartesian coordinates*

$$\begin{aligned} x' &= \{(\theta_1 - \theta_2 z^n)x - (1-\theta)\theta_1(x - yz^n)\}(\theta_1 - \theta_2 z^n)^{-1}, \\ y' &= \{(\theta_1 - \theta_2 z^n)y - (1-\theta)\theta_2(x - yz^n)\}(\theta_1 - \theta_2 z^n)^{-1}, \\ z' &= z, \end{aligned} \tag{5}$$

is birational. If V is a surface in $\mathbb{R}^3$ and the image $T_1(V)$ is a plane, then V is a rational ruled surface of degree $n+1$, and there is an unique generatrix of V in any plane P_t given by the equation $z - t = 0$ $(t \in \mathbb{R})$.

Proof: Substituting $X_1 X_0^{-1} = x$, $X_2 X_0^{-1} = y$ and $X_3 X_0^{-1} = z$ (into the equations (3)), we obtain (5). Then, the statements follow from Theorem 1 and Theorem 2. □

The inverse transformation T_1^{-1} is defined by (5), where θ is replaced by θ^{-1}. Hence, both T_1 and T_1^{-1} are not defined at the points of the reducible surface given by the equation $\theta_1 - \theta_2 z^n = 0$. Any surface V contains the reducible curve B_1 given by $\theta_1 - \theta_2 z^n = x - yz^n = 0$ which is the fundamental locus of T_1. If two lines are components of the curve B_1, then there is a family of rational ruled surfaces V passing through the lines. We shall use this property in the next sections.

Definition 2. *We say that the type of the surface $V \subset \mathbb{R}^3$ is $\mathcal{HP}_{n+1}$ if its transform $T_1(V)$ is a plane.*

It is clear that in the case $n = 1$, V is a hyperbolic paraboloid.

§4. Skew Lines

In this section we fix two skew lines L_1 and L_2 in the Euclidean space $\mathbb{R}^3$. Let φ be the angle between the lines L_1 and L_2, and let d be the distance between the same lines. Without loss of generality, we suppose that $0 < \varphi \leq \frac{1}{2}\pi$ and $\tan\varphi \neq d^n$. The mutual position of L_1 and L_2 is completely determined by φ and d.

Theorem 4. *For any positive integer n, there exists a two-parameter family of rational ruled surfaces which meet the following requirements:*

i) *The type of any surface is $\mathcal{HP}_{n+1}$,*

ii) *Any surface passes through L_1 and L_2.*

Proof: Let (x, y, z) be Cartesian coordinates in $\mathbb{R}^3$. Then, we may assume that

$$L_1 : \begin{cases} z - d & = 0 \\ x - d^n y & = 0 \end{cases} \quad \text{and} \quad L_2 : \begin{cases} z & = 0 \\ kx - y + 1 & = 0, \end{cases}$$

where $k = \frac{d^n \sin\varphi + \cos\varphi}{d^n \cos\varphi - \sin\varphi}$. Consider the birational transformation T_1 defined by (3) in which $\theta_1 = d^n$ and $\theta_2 = 1$. If the surface $V \subset \mathbb{R}^3$ is given as locus of

$$(d^n - z^n)(\lambda_0 + \lambda_1 x + \lambda_2 y + \lambda_3 z) - (1-\theta)(\lambda_1 d^n + \lambda_2)(x - yz^n) = 0, \quad (6)$$

where $\lambda_i \in \mathbb{R}$ $(i = 0,1,2,3)$ and $\sum_{i=0}^{3} |\lambda_i| \neq 0$, then the image $T_1(V)$ is a plane. Let $\lambda_0 = 1$, $\lambda_1 = k\,\theta^{-1} + (1-\theta^{-1})d^{-n}$, $\lambda_2 = -1$, and $\lambda_3 = \mu \in \mathbb{R}$. Thus, we obtain a 2-parameter family of rational ruled surfaces $V(\theta, \mu)$ given by

$$\begin{aligned}(d^n - z^n)\{1 + (\theta^{-1}k + \frac{1-\theta^{-1}}{d^n})x - y + \mu z\} \\ + (1-\theta^{-1})(d^n k - 1)(x - yz^n) = 0.\end{aligned}$$

The pencil of lines on $V(\theta, \mu)$ can be represented as $L(t) = P_t \cap Q_t$ $(t \in \mathbb{R})$, where the plane P_t is given by $z - t = 0$ and the plane Q_t is given by

$$\begin{aligned}(d^n - t^n)\{1 + (\theta^{-1}k + \frac{1-\theta^{-1}}{d^n})x - y + \mu t\} \\ + (1-\theta^{-1})(d^n k - 1)(x - yt^n) = 0.\end{aligned}$$

Then, $L_1 = L(t = d)$ and $L_2 = L(t = 0)$. □

Now, we can obtain a parametric representations of the above surfaces. If $\theta \neq 0, 1$ and μ are fixed real numbers, then the parametric equations of the surface $V(\theta, \mu)$ are

$$x = u \qquad y = \frac{f_1(u,t)}{g_1(t)} \qquad z = t,$$

where u and t are real parameters,

$$\begin{aligned}f_1(u,t) &= (t^d - d^n)\{1 + (\theta^{-1}\,k + d^{-n} - \theta^{-1}\,d^{-n})u + \mu t\} \\ &\quad + (1-\theta^{-1})(1 - k\,d^n)u \\ g_1(t) &= t^n - d^n + (1-\theta^{-1})(1 - k\,d^n)t^n.\end{aligned}$$

§5. Parallel Lines

Using the birational transformations defined in Section 3, we can construct noncylindrical rational ruled surfaces of odd degree which pass through two fixed parallel lines.

Theorem 5. *Let L_1 and L_2 be two parallel lines in the Euclidean space $\mathbb{R}^3$. Then, for any positive integer m, there exists a four-parameter family of rational ruled surfaces which meet the following requirements:*

i) *The type of any surface is $\mathcal{HP}_{2m+1}$;*

ii) *Any surface passes through L_1 and L_2.*

Proof: Let $2d$ be the distance between L_1 and L_2. Then, we may suppose that

$$L_j : \begin{cases} z + (-1)^j d & = 0 \\ z - d^{2m} y & = 0, \end{cases} \qquad j = 1, 2.$$

Consider the transformation T_1 given by the equations (3), in which $\theta_1 = d^{2m}$, $\theta_2 = 1$ and $n = 2m$. If $V \subset \mathbb{R}^3$ is a surface such that the image $T_1(V)$ is a plane, then V is given as locus of

$$(d^{2m} - z^{2m})(\lambda_0 + \lambda_1 x + \lambda_2 y + \lambda_3 z) - (1-\theta)(\lambda_1 d^{2m} + \lambda_2)(x - yz^{2m}) = 0,$$

where $\lambda_i \in \mathbb{R}$ and $\sum_{i=0}^{3} |\lambda_i| \neq 0$. For $t \in \mathbb{R}$, let P_t be the plane given by the equation $z - t = 0$ and Q_t be the plane given by the equation

$$(d^{2m} - t^{2m})(\lambda_0 + \lambda_1 x + \lambda_2 y + \lambda_3 t) - (1-\theta)(\lambda_1 d^{2m} + \lambda_2)(x - yt^{2m}) = 0.$$

Then the one-parameter family of lines $L(t) = P_t \cap Q_t \subset V$ contains the lines $L_1 = L(t = d)$ and $L_2 = L(t = -d)$. □

The above description of the generatrices of the surface V also gives its parametric equations

$$\begin{aligned} x &= u, \\ y &= \frac{(t^{2m} - d^{2m})(\lambda_0 + \lambda_1 u + \lambda_3 t) + (1-\theta)(\lambda_1 d^{2m} + \lambda_2)u}{\lambda_2(d^{2m} - t^{2m}) + (1-\theta)(\lambda_1 d^{2m} + \lambda_2)t^{2m}}, \\ z &= t, \end{aligned}$$

where u and t are real parameters.

Finally, we observe a special property of the surface $V \in \mathcal{HP}_{2m+1}$. From the proof of Theorem 5, it follows that the lines $L(t)$ and $L(-t)$ are parallel for any $t \neq 0$. Moreover, if $t_1 \neq t_2$ and $t_1 \neq -t_2$, then $L(t_1)$ and $L(t_2)$ are skew lines. In other words, the rational ruled surface $V \in \mathcal{HP}_{2m+1}$ is noncylindrical.

§6. Intersecting Lines

First, we consider another interpretation in $\mathbb{R}^3$ of the birational transformation T_0. Next, using this interpretation, we construct a four-parameter family of nonconic rational ruled surfaces which pass through two fixed intersecting lines.

Theorem 6. *Let ψ be an acute angle, and let $T_2 : \mathbb{R}^3 \to \mathbb{R}^3$ be the transformation given by the equations*

$$\begin{aligned} x' &= \frac{x\{h(x,y) - z^n\} - (1-\theta)\{xh(x,y) - yz^n\}}{h(x,y) - z^n - \sqrt{2}(1-\theta)\{xh(x,y) - yz^n\}\tan\psi} \\ y' &= \frac{y\{h(x,y) - z^n\} - (1-\theta)\{xh(x,y) - yz^n\}}{h(x,y) - z^n - \sqrt{2}(1-\theta)\{xh(x,y) - yz^n\}\tan\psi} \\ z' &= \frac{z\{h(x,y) - z^n\}}{h(x,y) - z^n - \sqrt{2}(1-\theta)\{xh(x,y) - yz^n\}\tan\psi}, \end{aligned} \tag{7}$$

where $\theta \in \mathbb{R} \setminus \{0,1\}$ and $h(x,y) = \{1 - \frac{\tan\psi}{\sqrt{2}}(x+y)\}^n$. Then, T_2 is birational. If $W \subset \mathbb{R}^3$ is a surface such that the image $T_2(W)$ is a plane, then W

is a rational ruled surface of degree $n+1$, and the singular locus of W is the $n-$fold line

$$S : \begin{cases} z & = 0 \\ x + y - \frac{\sqrt{2}}{\tan\psi} & = 0 \end{cases}.$$

Proof: Substituting into (3) $\theta_1 = \theta_2 = 1$, $\frac{X_1}{X_0+X_1+X_2} = \frac{x\tan\psi}{\sqrt{2}}$, $\frac{X_2}{X_0+X_1+X_2} = \frac{y\tan\psi}{\sqrt{2}}$, $\frac{X_3}{X_0+X_1+X_2} = z$, we get (7). Thus, the statement follows from Theorem 1 and Theorem 2. □

Definition 3. *We say that the type of a surface $W \subset \mathbb{R}^3$ is $\mathcal{HOS}_{n+1}$, if the transform $T_2(W)$ is a plane.*

Note that in the case $n = 1$, W is a hyperboloid of one sheet.

Theorem 7. *Let L_1 and L_2 be two intersecting lines in $\mathbb{R}^3$, and let m is a positive integer. Then, there is a four-parameter family of rational ruled surfaces such that the type of any surface is $\mathcal{HOS}_{2m+1}$ and any surface passes through L_1 and L_2.*

Proof: Let 2ψ be the angle between L_1 and L_2. Then, we may suppose that

$$L_j : \begin{cases} x - y & = 0 \\ (x+y)\tan\psi + (-1)^{j+1}\sqrt{2}\,z - \sqrt{2} & = 0 \end{cases} \quad j = 1, 2.$$

Consider the transformation T_2 given by the equation (7) in which $n = 2m$. If $W \subset \mathbb{R}^3$ is a surface and the image $T_2(W)$ is a plane, then W is given by

$$\begin{aligned} &(\lambda_0 + \lambda_1 x + \lambda_2 y + \lambda_3 z)\{h(x,y) - z^n\} \\ &- (1-\theta)(\sqrt{2}\,\lambda_0 \tan\psi + \lambda_1 + \lambda_2)\{xh(x,y) - yz^n\} = 0, \end{aligned} \tag{8}$$

where $\lambda_i \in \mathbb{R}$, $\sum_{i=0}^{3} |\lambda_i| \neq 0$. Let $\overline{P}_t \quad (t \in \mathbb{R})$, be a plane given by

$$1 - \frac{\tan\psi}{\sqrt{2}}(x + y) = 0 \tag{9}$$

and let $\overline{Q}_t \quad (t \in \mathbb{R})$ be a plane given by

$$\begin{aligned} &(\lambda_0 + \lambda_1 x + \lambda_2 y + \lambda_3 t)(1 - t^{2m}) \\ &- (1-\theta)(\sqrt{2}\,\lambda_0 \tan\psi + \lambda_1 + \lambda_2)\{x - y\,t^{2m}\} = 0. \end{aligned} \tag{10}$$

Then, the line $L(t) = \overline{P}_t \cap \overline{Q}_t$ lies on W for any $t \in \mathbb{R}$. It is easy to see that $L_1 = L(t = 1)$ and $L_2 = L(t = -1)$. Hence, W is a ruled surface which contains L_1 and L_2. □

Equation (8) with the additional conditions $x \leq 0$ and $y \leq 0$ determines a domain on the surface W which is smooth and without singularities. Other properties of W follow from (9) and (10). The line $L(t)$ intersects the line $L(-t)$ at a point on $n-$fold line S for any $t \in \mathbb{R} \setminus \{0\}$. If $t_1 \neq t_2$ and $t_1 \neq -t_2$, then $L(t_1)$ and $L(t_2)$ are skew lines. This means that W is a nonconical surface.

Finally, the parametric representation of the ruled surface W is

$$x = u, \qquad y = \frac{f_2(u,t)}{g_2(t)}, \qquad z = t\{1 + \frac{\tan\psi}{\sqrt{2}}(u + \frac{f_2(u,t)}{g_2(t)})\},$$

where u and t are real parameters, and

$$f_2(u,t) = (1-\theta)(\sqrt{2}\tan\psi\,\lambda_0 + \lambda_1 + \lambda_2) - (\lambda_0 + \lambda_1\,u + \lambda_3\,t)(1 - t^{2m}),$$
$$g_2(t) = (1-\theta)(\sqrt{2}\tan\psi\,\lambda_0 + \lambda_1 + \lambda_2)\,t^{2m} + \lambda_2(1 - t^{2m}).$$

Using the general properties of the birational transformations, we get the implicit equations of the surfaces in the last three sections. But to find the parametric representations of these surfaces, we apply the specific properties of the transformations T_1 and T_2.

References

1. Abhyankar S. S., *Algebraic Geometry for Scientists and Engineers*, AMS, Providence R.I., 1990.
2. Chen, H.-Y., and H. Pottmann, Approximation by ruled surfaces, J. Comput. Appl. Math. **102** (1999), 143–156.
3. D'Almeida, J., Lien singulier d'une surface réglée, Bull. Soc. Math. Fr. **118** (1990), 395–401.
4. Mamford, D., *Algebraic Geometry I, Complex Projective Varieties*, Springer-Verlag, Berlin, 1995.
5. Pottmann, H., W. Lü, and B. Ravani, Rational ruled surfaces and their offsets, Graphical Models and Image Processing **58** (1996), 544–552.
6. Ravani, B., and J. Wang, Computer aided geometric design of line constructs, ASME J. Mech. Design **113** (1991), 361–371.
7. Semple, J. G., and G. T. Kneebone, *Algebraic Projective Geometry*, Clarendon Press, Oxford, 1998.

Gueorgui H. Gueorguiev
Faculty of Mathematics and Informatics
Shumen University
9712 Shumen, Bulgaria
g.georgiev@shu-bg.net

Deformation Criteria for the Direct Manipulation of Free Form Surfaces

Stéphane Guillet and Jean-Claude Léon

Abstract. The approach proposed here is based on coupling a mechanical model to the input surface geometry provided by the designer. The mechanical model is based on a set of bar networks subjected to tension forces only. Constraints are specified by the designer to express the deformation behaviour of the surface in the area of interest and define functional dimensioning objectives to perform a direct manipulation of the surface. Generally, this process leads to a globally underdetermined system of equations, i.e. the number of unknowns (external forces) is significantly greater than the number of equations generated by the designer's constraints. To this end, a minimization problem is formulated which expresses various deformation behaviours. In contrast to difference of deformation approaches based on mechanical models like membrane models, finite element models, which solely rely on strain energy minimization criteria, the approach proposed here provides the designer various criteria to help him/her create different deformation behaviours like an area minimizing criterion, expressing a minimum change of the shape in the deformed area, expressing a deformation with slowly varying curvature in the deformed area, providing a deformation behaviour which allows to approximately preserve the section of pipe-like surfaces subjected to bending deformations. As depicted, multiple criteria help the designer express various deformation behaviours which are required during a design process.

§1. Context of Surface Deformations

The shape modification of an object during a design process depends on the context of this process. In the field of mechanical engineering design, constraint requirements can be either aesthetic [3,7] or functional [1].

Without adequate 3D modification tools, the surface deformation leads the designer to tedious manipulations, i.e., displacements of numerous control polyhedron vertices, chain modifications of patches or surfaces, etc.

The basic aim of these deformation tools would be to provide the user an easy and intuitive control of the surface shape. Their parameters should be automatically related to the parameters governing the deformation process.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 135–144.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

Such approaches fit into a class [3,5,9] that helps a designer shape the overall object, but they are not suitable for generating free-form surfaces which accurately match geometric constraints involving functional parameters. A second class of 3D modification tools covers the approach of Celniker [2], Kondo [6], Light [8] and Welch [10]. These types of tools fall into the domain of parametric or variational design tools. Among these approaches, some [6,8] focus on parametric or variational models applied to 2D models. Others [2,10] perform a surface deformation subjected to constraints such as prescribed curvature or surface rectitude using a membrane model which cannot provide some deformation modes like bending.

The approach presented here fits into this last category. Similar to the approach of Celniker and Welch, the current one also uses a mechanical model. However, its formulation is simpler than Celniker and Welch's, and thus it is easier to manipulate and it allows generation of isotropic and anisotropic deformations. The approach introduced here is a new development around a free-form deformation method [4]. The scope of the present work focuses on the introduction of a set of deformation criteria which cannot be provided by membrane models or other mechanical models subjected to small displacements and linear behaviour material law hypotheses.

§2. Principle of the Parametric Deformation

Before studying the deformation criteria presented here, it is suitable to summarize the objectives and the constraints related to the parametric deformation process. The features of the current work are the following:

- the treatment of configurations involving multiple trimmed free-form surfaces based on a B-Spline model,
- the direct manipulation of the geometry through a small number of parameters to allow an easy and intuitive control of the surface shape,
- the possibility for the user to create local or global deformations of the geometry and to obtain different solutions with one set of geometric constraints,
- the fast computation which allows an easy integration of the parametric deformation tool into an integrated design process.

In the context of the approach introduced here, different constraints reduce the complexity of the problem:

- the surface patch decomposition is preserved, i.e., degrees, nodal sequences and topology are kept constant,
- C^0 continuity between patches is maintained. G^0 and G^1 continuities are approximated along the trimming lines. To this end, a discretization process is applied to these lines without modification of their degree,
- trimming lines on the surface are kept unchanged into their parametric space. A trimming line is defined as a set of connected trimming curves.

The aim of the parametric approach is to deform a set of trimmed free-form surfaces subject to geometric constraints. The geometric constraints are cur-

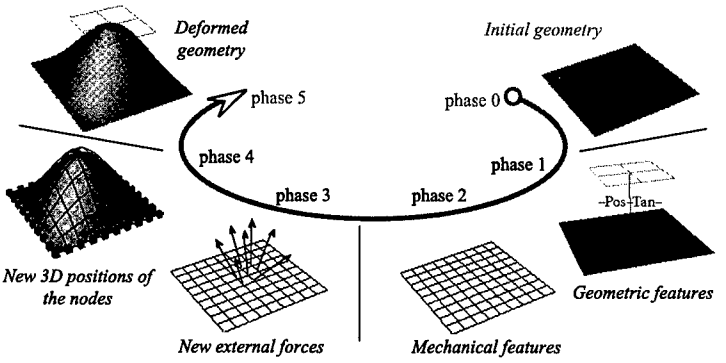


Fig. 1. Main steps of the parametric modelling process.

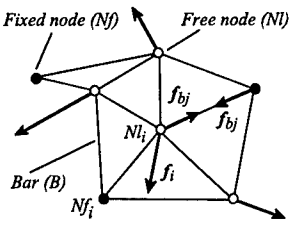
rently restricted to the control of the dimensions of an object though other categories of constraints can be set up to achieve other functions.

Prior to the description of the minimization criteria used here, it is suitable to describe how geometric and mechanical features fit together. The resulting surface geometry is obtained from the initial one through five steps (Figure 1):

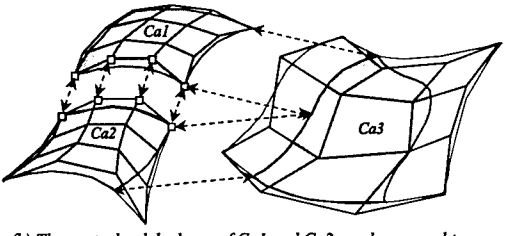
- step one is devoted to the creation of the geometric features. These features help specify some target parameters of the surface shape, the deformed area and the continuity conditions between trimmed patches. In the case of Figure 1, one feature is generated by the user. Currently, the design constraints can be displacements of points, lines; tangency constraints with planes; contact with another free form surface; internal continuity constraints between patches,
- step two of this method involves mechanical features. These features are based on parameters of the mechanical model (topology, mobility and force density) used to obtain a deformed geometry and on the choice of a minimization criterion. Thus, the user can obtained different solutions with a unique set of geometric features.

These first two steps are devoted to the initialization of the process. The user can modify one or all these features if he/she does not accept the deformed geometry. The next two steps are transparent for the user, and focus on the computation of the deformed geometry:

- step three: a relationship between geometric and mechanical features contributes to the computation of new external forces through an optimization process. Different minimization functionals can be incorporated into this process,
- step four: these new forces influence the static equilibrium positions of bar networks (mechanical models). New 3D positions of the nodes of the bar networks are computed, i.e., new 3D positions of the vertices of the control polyhedrons of the trimmed surfaces.



(a) An example of bar network



(b) The control polyhedrons of Ca1 and Ca2 can be merged to form onebar network because they are connected along their entire boundary whereas Ca3 generates another bar network since Ca3 is connected to Ca1 and Ca2 along a trimming line.

Fig. 2. Bar networks used to control the shape of surfaces.

§3. Mechanical Deformation Features

Mechanical model of deformation

The bar network (Figure 2a) is built from bars B with pin joints which are assumed to rotate without friction [4,11]. All bars are under tension. The parameters governing the static equilibrium state of such a network are

- the *mobility* of the nodes N, i.e., fixed (N_f) or free (N_l) to move in 3D space,
- the *topology* of the bar network, i.e., the way the bars B are connected to the nodes N of the network,
- the *force density* q_j attached to each bar B_j of the network is defined as the ratio between the internal force $\mathbf{f}_{bj}$ into the bar and its length l_j ($q_j = \frac{\|\mathbf{f}_{bj}\|}{l_j}$, $q_j > 0$). The positivity constraint ensures the tension state in every bar B_j,
- the *external force* $\mathbf{f}_i$ which may be applied to the i^{th} mobile node of the bar network.

Linear static equilibrium

Static equilibrium of a bar network is achieved when the sum of the external force $\mathbf{f}_i$ applied at the i^{th} node equilibrate the forces applied by each bar meeting at that node. This statement becomes

$$\mathbf{f}_i + \sum_{j=0}^{n_{bi}} q_j.(\mathbf{x}_k - \mathbf{x}_{li}) = \mathbf{0}, \qquad i \in \{1, \ldots, n_l\}, \tag{1}$$

where $\mathbf{x}_{li}$ stands for the coordinates of the i^{th} free node, $q_j.(\mathbf{x}_k - \mathbf{x}_{li})$ denotes the tension force into the j^{th} bar meeting at node i, n_{bi} designates the number of bars meeting at node i and n_l the number of free nodes of the bar network.

Given the topology of the bar networks, the equilibrium equations are

$$\begin{cases} \mathbb{D}_l \,.\, \mathbf{x}_l + \mathbb{D}_f \,.\, \mathbf{x}_f = \mathbf{f}_x, \\ \mathbb{D}_l \,.\, \mathbf{y}_l + \mathbb{D}_f \,.\, \mathbf{y}_f = \mathbf{f}_y, \\ \mathbb{D}_l \,.\, \mathbf{z}_l + \mathbb{D}_f \,.\, \mathbf{z}_f = \mathbf{f}_z. \end{cases}$$

for one bar network, where $\mathbb{D}_l$ is a positive definite matrix which ensures the existence and unicity of either equilibrium positions ($\mathbf{x}_l$, $\mathbf{y}_l$, $\mathbf{z}_l$) or external forces ($\mathbf{f}_x$, $\mathbf{f}_y$, $\mathbf{f}_z$).

Generally, the objective is to calculate the 3D position of free nodes. These new positions are obtained with a linear equation system through changes of variables of the parametric process, i.e., the external forces $\mathbf{f}_i$.

Analogy used for surface deformation

The deformation method uses an analogy between the control polyhedron of a surface and the mechanical equilibrium position of a bar network. One bar network is associated with one or several surfaces as follows:

- the nodes of a bar network coincide with the entire set of vertices of the control polyhedron of a free-form surface (either trimmed or not),
- the C^0 continuities are directly incorporated into the mechanical model. In this case, several bar networks can be merged together.

§4. Deformation Criteria

The geometric constraints generated by the designer to prescribe dimensions combined with the equilibrium equations of the bar networks form the global set of constraints

$$\begin{aligned} \mathbf{G} &= \mathbf{G}(f_{1x}, \ldots, f_{n_l x}, f_{1y}, \ldots, f_{n_l y}, f_{1z}, \ldots, f_{n_l z}), \\ &= \mathbf{G}(\mathbf{F}) = \mathbf{0}, \ i \in \{1, \ldots, n_c\}, \end{aligned} \tag{2}$$

where each constraint is expressed in terms of the external forces applied to the bar networks. External forces have been chosen as unknowns rather than force densities q_j because they produced intermediate solutions which reflected a real deformation process of a surface whereas iterating with q_j produced oscillations around the solution. Then, the resolution has been conducted using an augmented Lagrangian method to provide robustness to the solving process.

Because the number n_c of constraints is usually significantly smaller than the number of unknowns, and assuming that there is no local configuration with an overconstrained subset of equations, a functional $\Phi(\mathbf{F})$ can be associated with (2) to obtain a solution which matches a specific designer's interest. Overconstrained subsets of equations can be encountered when a subset of (2) is such that there exists locally for a given surface more constraints then the

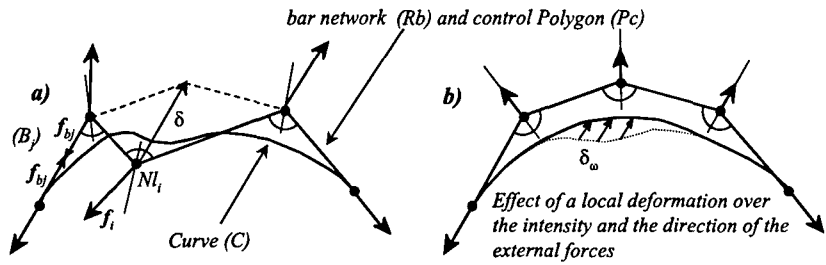


Fig. 3. Relationships between external forces at nodes and the shape of a curve.

number of free parameters x_l, y_l, z_l. When $\mathbf{G}$ contains nonlinear equations, their derivatives are related to the geometrical and mechanical parameters by

$$G_{ij}^{[k]} = \frac{\partial G_i}{\partial F_j} = \frac{\partial G_i}{\partial X_p} \cdot \frac{\partial X_p}{\partial x_{l_q}} \cdot \frac{\partial x_{l_q}}{\partial F_j} + \frac{\partial G_i}{\partial Y_p} \cdot \frac{\partial Y_p}{\partial y_{l_q}} \cdot \frac{\partial y_{l_q}}{\partial F_j} + \frac{\partial G_i}{\partial Z_p} \cdot \frac{\partial Z_p}{\partial z_{l_q}} \cdot \frac{\partial z_{l_q}}{\partial F_j}, \quad \forall i.$$

where $\frac{\partial G_i}{\partial X_p}, \frac{\partial G_i}{\partial Y_p}, \frac{\partial G_i}{\partial Z_p}$ are related to the geometric constraints set by the designer, $\frac{\partial X_p}{\partial x_{l_q}}, \frac{\partial Y_p}{\partial y_{l_q}}, \frac{\partial Z_p}{\partial z_{l_q}}$ come from the relationship between the surface and the bar networks and $\frac{\partial x_{l_q}}{\partial F_j}, \frac{\partial y_{l_q}}{\partial F_j}, \frac{\partial z_{l_q}}{\partial F_j}$ are coefficients of $\mathbb{D}_l^{-1}$.

External forces at nodes and shape relationships

According to (1), external forces at nodes are governed by the length of the bars as well as the angle between the bars meeting at a node. Figure 3a illustrates such a configuration for a bar network which corresponds to the control polygon of a Bézier curve with uniform force densities in its bars.

Then, it can be stated that a regular bar network has *smaller external forces* than an irregular one, since regular control polygons resemble the shape of the curve, and therefore have smaller length sides and wider angles between bars (Fig. 3b).

With uniform force densities, the direction of the external forces is close to the bisecting line of two adjacent bars or, for a bar network attached to a surface patch, this direction is close to the average normal direction at the given node. However, this behaviour does not necessarily generate acceptable shapes (as depicted in Figure 4) and needs to be combined with the regularity criterion of the intensities.

A change in the intensity of an external force at a node N_k results in a change of position of the free nodes whose amplitude decrease from N_k. The direction of movement of the free nodes is similar to that of the external force which has been modified [7] at N_k. In turn, the displacement of the points on a curve or on a patch follows the geometric property of Bézier or B-Spline models, i.e., points move in the direction of the movement of a control point. Their displacement amplitude is therefore smaller than that of N_k.

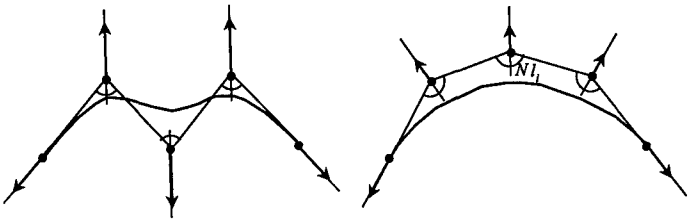


Fig. 4. Specific configuration where external forces coinciding with bisecting lines does not provide a smooth curve.

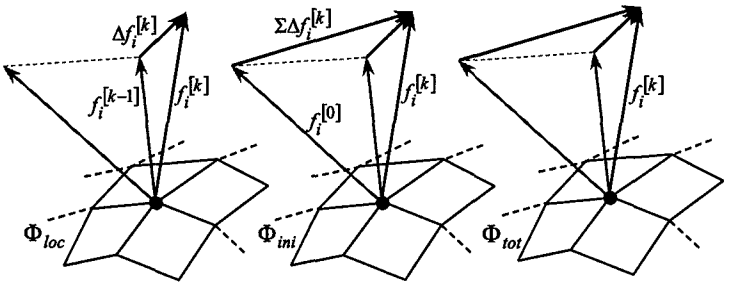


Fig. 5. Criteria related to the external force at a free node.

Various deformation criteria

Based on the previous relationships, three categories of criteria have been identified:

- the first one is related to the external forces at the free nodes of the bar networks,
- the second one acts simultaneously over the external forces and average normal direction at the free nodes,
- the third one acts over the average force around a free node, but is not described here because of lack of space).

The first category takes as input either the external force $\mathbf{f}_i^{[k]}$ or the variation $\Delta\mathbf{f}_i^{[k]}$of this force at the iteration $[k]$ when the constraints expressed are nonlinear (see Figure 5).

When considering the functional $\Phi_{loc}^{[k]} = \min\left({}^T\Delta\mathbf{F}^{[k]}.\Delta\mathbf{F}^{[k]}\right)$, the designer expresses the minimum change in the shape of the object at each iteration until the constraints $\mathbf{G}$ are satisfied.

When considering $\Phi_{ini}^{[k]} = \min\left({}^T(\mathbf{F}^{[k]} - \mathbf{F}^{[0]}).(\mathbf{F}^{[k]} - \mathbf{F}^{[0]})\right)$ as the functional associated with $\mathbf{G}$, the designer expresses the minimum change of the object shape between the input geometry and the output. In case of linear constraints, $\Phi_{loc}^{[k]}$ and $\Phi_{ini}^{[k]}$ are identical. This functional tends to preserve as much as possible the previous work of the designer, and therefore is of specific interest during a modeling process.

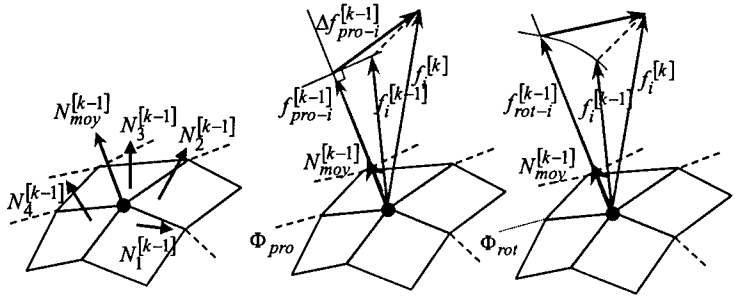


Fig. 6. Criteria acting over the intensity and the direction of the external force at a node.

When considering the functional $\Phi_{tot}^{[k]} = \min\left({}^T\mathbf{F}^{[k]}.\mathbf{F}^{[k]}\right)$, the designer expresses the fact that the resulting surface is not based on the input one since the initial external forces are not taken into account. Furthermore, the minimization of the intensity of the external forces at the free nodes expresses that the output control polyhedrons form an approximation of a minimal surface. Hence, the output surface attached to the bar networks represents an approximation of the minimum surface area satisfying the constraints. This approximation is even more effective when the control polyhedrons converge toward the surface itself.

The second category of criteria acts simultaneously over the intensity and direction of the external forces at free nodes so that the direction of the forces meet a given criterion. To this end, an average normal direction is built according to the position of the nodes around the target node. Figure 6 illustrates the planes surrounding the i^{th} free node at iteration $[k-1]$ which participate to the definition of the average normal direction $\mathbf{N}_{moy-i}^{[k-1]}$. When considering the functional $\Phi_{pro}^{[k]} = \min\left({}^T\Delta\mathbf{F}_{pro}^{[k]}.\Delta\mathbf{F}_{pro}^{[k]}\right)$, the forces minimized correspond to the difference between the projection $\mathbf{f}_{pro-i}^{[k-1]}$ of the external force $\mathbf{f}_i^{[k-1]}$ onto $\mathbf{N}_{moy-i}^{[k-1]}$ and the force $\mathbf{f}_i^{[k]}$ at the k^{th} iteration, i.e. $\Delta\mathbf{F}_{pro-i}^{[k]} = \mathbf{f}_i^{[k]} - \frac{\mathbf{f}_i^{[k-1]}.\mathbf{N}_{moy-i}^{[k-1]}}{\|\mathbf{f}_i^{[k-1]}.\mathbf{N}_{moy-i}^{[k-1]}\|}.\mathbf{f}_i^{[k-1]}$.

Using this criterion, the designer expresses that the output surface tends to minimize the area while being smooth since the intensity of the forces tend to decrease like $\Phi_{tot}^{[k]}$ as well as the direction of the forces tend to be more regular using $\mathbf{N}_{moy-i}^{[k-1]}$. This criterion takes into account the shape of the input geometry, but generates a surface which is 'smoother' than with $\Phi_{ini}^{[k]}$.

When considering the functional $\Phi_{rot}^{[k]} = \min\left({}^T\Delta\mathbf{F}_{rot}^{[k]}.\Delta\mathbf{F}_{rot}^{[k]}\right)$, the forces minimized correspond to the difference between the external force $\mathbf{f}_i^{[k-1]}$ rotated onto the direction of $\mathbf{N}_{moy-i}^{[k-1]}$ and the force $\mathbf{f}_i^{[k]}$ at the k^{th} iteration, i.e. $\Delta\mathbf{F}_{rot-i}^{[k]} = \mathbf{f}_i^{[k]} - \|\mathbf{f}_i^{[k-1]}\|.\mathbf{N}_{moy-i}^{[k-1]}$. Such a functional tends to preserve

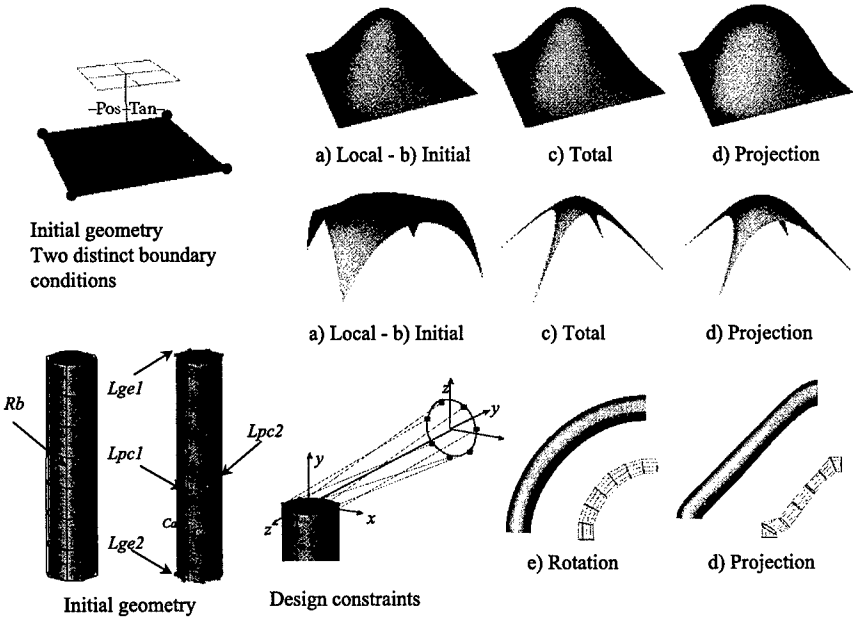


Fig. 7. Illustration of the influence of the deformation criteria.

the intensity of the external forces while modifying their direction in order to smooth the surface. Combined with anisotropic force densities in the bars of the networks, this criterion expresses the deformation behaviour of pipe-type objects when their are subjected to bending deformations. Though the previous criteria are nonlinear even if $\mathbf{G}$ is linear, their efficiency is strong enough to justify their use during a design process.

§5. Results and Examples

The above deformation criteria have been applied to different categories of surfaces to illustrate their typical behaviour according to the configurations described in the previous section. Figure 7a shows the effect of Φ_{loc}, 7b illustrates Φ_{ini}, 7c Φ_{tot}, 7d Φ_{pro} and 7e Φ_{rot} under various designer constraints.

Two distinct input geometries are used. The upper one is a one patch surface, and two types of boundary conditions were used, i.e. fixed boundary lines and fixed corner points. The design constraint is formed by a position and tangency constraint. The bottom one is a multipatch surface with G^1 continuity constraints, where the designer has specified position and tangency constraints along the extreme boundary line of the surface.

§6. Concluding Remarks

The deformation criteria presented provide a diversity of control of shape. They form an efficient complement to the geometric constraints set by the de-

signer to let him/her adapt the result to his/her needs. Such a diversity cannot be achieved using a mechanical approach solely based on a minimization of the strain energy of membrane type structures. Furthermore, the criteria set up are not bound by a small displacements hypothesis and can handle geometric constraints involving significant shape changes.

Future work will focus on the cross influence between the deformation criteria and the boundary conditions applied to the bar networks to provide more intuitive user interactions.

References

1. Cavendish, J. C., Integrated feature-based surface design with free-form deformation, Computer-Aided Design **27**, n°9 (1995), 703-711.
2. Celniker, C., and W. Welch, Linear constraints for deformable B-splines surfaces, Proceedings of the 1992 Symposium on Interactive 3D Graphics (1992), 165–170.
3. Coquillart, S., Extended free-form deformation: A sculpturing tool for 3D geometric modelling, SIGGRAPH 90 **24** (1990), 187–196.
4. Guillet, S., and J.-C. Léon, Parametrically deformed free-form surfaces as part of a variational model, Computer-Aided Design **30**, n°8 (1998), 621–630.
5. Kang, H., and A. Kak, Deforming virtual objects interactively in accordance with an elastic model, Computer-Aided Design **28**, n°4 (1996), 251–262.
6. Kondo, K., PIGMOD: parametric and interactive geometric modeller for mechanical design, Computer-Aided Design **22**, n°10 (1990), 633–644.
7. Léon, J. C., and P. Trompette, A new approach towards free-form surfaces control, Comput. Aided Geom. Design **12** (1995), 395–416.
8. Light, R., and D. Gossard, Modification of geometric models through variational geometry, Computer-Aided Design **14**, n°4 (1982), 209–214.
9. Sederberg, T. W., and S. R. Parry, Free form deformation of solid geometric models, SIGGRAPH 86 **20** (1986), 151–160.
10. Welch, W., and A. Witkin, Variational surface modelling, SIGGRAPH 92 **26** (1992), 157–166.
11. Schek, H.J., The force density method for form finding and computation of general networks, Comp. Meth. in App. Mech. and Eng. **3** (1974), 115-134.

Jean-Claude Léon and Stéphane Guillet
Laboratoire 3S, Domaine Universitaire, BP 53X
38041 GRENOBLE Cedex 9 FRANCE
Jean-Claude.Leon@hmg.inpg.fr
Stephane.Guillet@hmg.inpg.fr

Generation of Surfaces with Smooth Highlight Lines

Masatake Higashi, Hiroto Harada,
and Mitsuru Kuroda

Abstract. This paper proposes a method which generates smooth surfaces from four boundary curves. A criterion is introduced to represent smoothness of highlight lines which are an approximation of reflection lines and are sensitive to the surface irregularity. The criterion is the square of projected curvature of highlight lines per unit length. To obtain the surfaces which satisfy the criterion, the evolutes of their parametric lines which influence highlight lines are determined to change smoothly. The evolutes are represented with two segments of second-degree rational Bézier curves, whose parameters are determined to minimize the criterion. The method is extended to determine a surface when a highlight line is given by a designer. Some examples of boundary curves with various patterns of curvature variation are shown to generate smooth surfaces.

§1. Introduction

In the design of aesthetic shapes like automotive bodies, curvature variation of surfaces is very important. Designers determine shapes according to their great concern for the reflected images of the surroundings, shade lines, and highlight lines. Since reflection and shading are affected by changes of surface normal, the curvature distribution of the surface should be smooth and formed as designers want.

Spline interpolation, fairing and lofting methods [3,8] which are widely employed in industrial applications cannot assure smooth distribution of curvature of a surface, although they generate surfaces which pass through the given points and satisfy second degree continuity. On the other hand, methods [4,10] which simulate the minimization of the elastic energy for a thin plate can generate smooth surfaces, but cannot always obtain the shape which designers want for the given boundary conditions.

Hence we have proposed a surface generation method [7] which directly determines curvature distribution of a surface from four boundary curves by smoothly interpolating the locus of an *evolute* of a generatrix of a surface. In this paper, we extend the method to determine a shape which has smooth highlight lines directly according to the criterion introduced.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 145–152.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

§2. Highlight Lines and their Criterion

Highlight lines are images (reflection lines) on surfaces of a product or its clay model for parallel lines such as fluorescent lamps on a ceiling, and they are used for the evaluation of surfaces in the automotive industry. If they are not as smooth as designers want, the surface of the clay model is modified until the shape becomes satisfactory.

To evaluate and modify these images in the computer using a CAD system, several methods were proposed. Klass [9] tried to correct local irregularities of a surface using reflection lines. Chen and Beier [1,2] introduced an equation of approximated highlight lines for the real time evaluation of a surface, and applied it to modification of NURBS surfaces. The equation represents normal projection of parallel lines to surfaces. On the other hand, Higashi et al. [5,6] introduced an equation of pseudo-highlight lines which are *silhouette lines* of a surface for incident directions. These highlight lines have been used in a practical CAD system in the automotive industry [5], because they are sensitive to surface irregularities and they had been checked manually on drawings.

Let an incident direction, a surface, and its normal at parameter (u, v) be $\boldsymbol{L}$, $\boldsymbol{S}(u, v)$ and $\boldsymbol{n}(u, v)$. The equation of a silhouette line is

$$\boldsymbol{n}(u, v) \cdot \boldsymbol{L} = 0. \tag{1}$$

If the incident direction is rotated around an axis, we obtain a group of silhouette lines and call them a silhouette pattern.

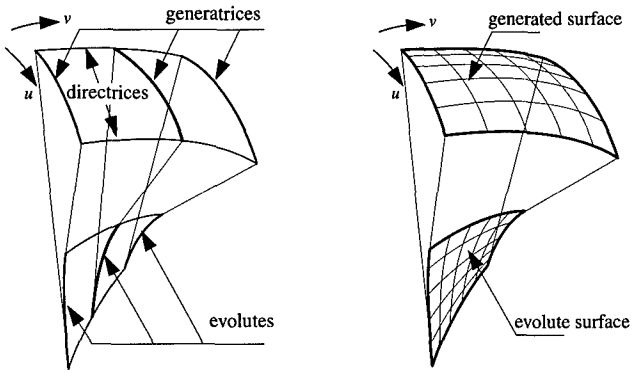
We introduce a criterion H of smooth highlight lines, that is a silhouette pattern, in order to automatically generate a surface which designers want. Let the projected curvature of a silhouette line be $\kappa_i(s)$. Here, suffix i corresponds to highlight line i, s is a parameter of an arclength, and the number of highlight lines is n. We denote the length of each line by s_i. Then we get

$$H = \sum_{i=1}^{n} \int_0^{s_i} \kappa_i^2(s) ds \Big/ \sum_{i=1}^{n} s_i. \tag{2}$$

§3. Concept of Surface Generation Based on Evolute

A surface is generated by moving a generatrix along two directrices. When the shape of the generatrix is changed with movement, the interpolation of the movement is not simple. Blending of boundary curves or interpolation of the boundary conditions does not necessarily create a good curvature distribution of the surface.

A generatrix should be moved so that the curvature distribution becomes smooth and satisfies the highlight line criterion described in the previous section. The curvature distribution of the surface is represented as a surface, by making the locus of the evolute of the generatrix. We call the surface generated from the locus an evolute surface. Fig. 1(a) shows an object surface



(a) Generatrix and evolute (b) Evolute surface and generated surface

Fig. 1. Generatrix and evolute surface.

and its generatrices along with the corresponding evolutes. If the generatrix is a space curve, its evolute cannot be determined uniquely. We define the evolute to be related to the surface property by fixing the freedom around the tangential direction [7]. Let the given curve, the curvature radius and torsion be $\boldsymbol{R}(s)$, $\rho(s)$ and $\tau(s)$, and let $\boldsymbol{n}(s)$ and $\boldsymbol{b}(s)$ denote normal and binormal vectors. They are represented as functions of arclength s. Then the equation of the evolute is

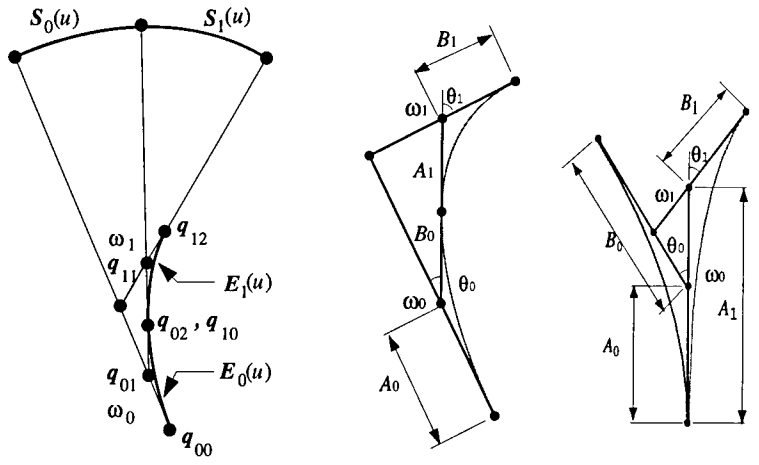
$$\boldsymbol{r}(s) = \boldsymbol{R}(s) + \rho(s)\{\boldsymbol{n}(s) + \tan(-\int \tau(s)ds + \Phi)\boldsymbol{b}(s)\}. \tag{3}$$

We determine the arbitrary constant Φ so that the starting point of the evolute is located at the direction of the surface normal, defined by the outer product of the tangents of the generatrix and the directrix.

Since an evolute is a curve of the curvature center of a generatrix, the quality of the surface is satisfactory if its evolute surface is smooth. Hence we determine the evolute surface first as a smooth surface, and then we align it according to the constraints of the evolute such that the difference of the curvature radii at the end points is equal to the length of the evolute and the tangent directions at the end points of the evolute are the same with those of the normal vectors of the involute. We note that a generatrix corresponds to a v-constant parametric line of the surface, and is represented by parameter u. Let the generated surface (involute), the evolute surface and the curvature radius at the starting point be $\boldsymbol{S}(u,v), \boldsymbol{E}(u,v)$ and $\rho(v)$. Then we get the equation of the object surface:

$$\boldsymbol{S}(u,v) = \boldsymbol{E}(u,v) + \{\rho(v) - \int_0^u |\boldsymbol{E}_u(u,v)|du\}\frac{\boldsymbol{E}_u(u,v)}{|\boldsymbol{E}_u(u,v)|}. \tag{4}$$

Here, the curvature radius $\rho(v)$ is determined by aligning the evolute to the directrices, and suffix u denotes partial differentiation. Fig. 1(b) shows an evolute surface and a generated surface satisfying the constraints.



(a) Control points of evolute (b) Shape parameters

Fig. 2. Control points and shape parameters of evolute.

§4. Surface Generation Satisfying Highlight Line Criterion

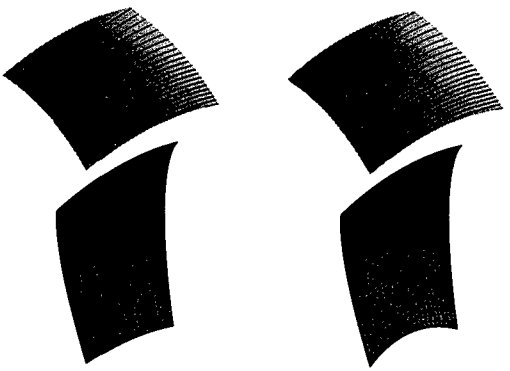
We generate a surface with smooth highlight lines as well as smooth curvature distribution. We approximate the evolute of the generatrix by second-degree rational Bézier curves because they are conics and have smooth curvature distribution. Then, we interpolate an evolute surface smoothly from the evolutes of two boundary curves. If we interpolate the shapes of the evolutes linearly, the surface becomes smooth, but the highlight lines do not necessarily satisfy designers. So, we interpolate the change of the shape of the evolute using a polynomial function.

We approximate an evolute with two segments of Bézier curves as shown in Fig. 2(a). By using two segments, we can represent all the patterns of curvature distribution of a simple curve. The patterns are divided into monotone (increasing or decreasing) and not monotone (with maximum or minimum in the middle). When a curve is not monotone in curvature, its evolute has a cusp point in the middle. Fig. 2(b) shows examples of evolutes and their Bézier polygons for different curvature patterns. The left figure has a monotone curvature distribution, and the right figure has a cusp point at the maximum curvature radius. We connect two segments at the junction point with tangential continuity. Control points $\mathbf{q}_{01}, \mathbf{q}_{02}(\mathbf{q}_{10})$, and $\mathbf{q}_{11}$ are collinear.

Interpolating control points $\mathbf{q}_{i0}, \mathbf{q}_{i1}, \mathbf{q}_{i2}$, and weight ω_i along v direction, we get an evolute surface

$$\mathbf{E}_i(u,v) = \frac{B_0^2(u)\mathbf{q}_{i0}(v) + B_1^2(u)\omega_i(v)\mathbf{q}_{i1}(v) + B_2^2(u)\mathbf{q}_{i2}(v)}{B_0^2(u) + B_1^2(u)\omega_i(v) + B_2^2(u)}. \tag{5}$$

Here, the index i represents the i-th segment, and $B_j^2(u)$ is a Bernstein polynomial.



(a) (b)

Fig. 3. Example 1 of smooth highlight lines. (a) linear interpolation: $H = 0.128 \times 10^{-3}$, (b) minimum H: $H = 0.055 \times 10^{-3}$.

Since we interpolate the shape of the evolute instead of the positions of control points, the number of its independent parameters becomes eight as shown in Fig. 2(b). They are the lengths of edges (A_i, B_i), the included angles θ_i between edges, and the weights ω_i, $i = 1, 2$. Using the shape parameters, we align the control polygons on the directrices. Then we get the functions of control points in (5).

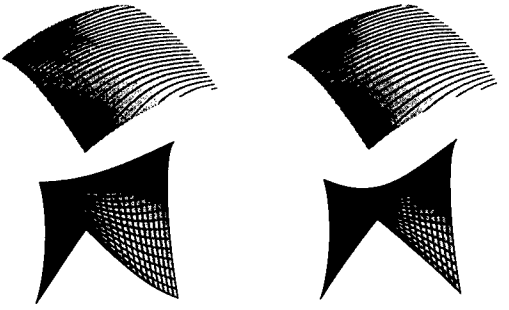
We interpolate these shape parameters smoothly with a second-degree polynomial. Let the set of the parameters be λ. Then we get

$$\lambda(v) = (1-v)^2\lambda(0) + 2v(1-v)\lambda_c + v^2\lambda(1). \tag{6}$$

λ_c is a control variable for each shape parameter. We determine these control variables to obtain a surface with smooth highlight lines by minimizing eq. (2). Starting from the values of λ for the linear interpolation, we search the values for the minimum criterion by changing them so as to decrease H step by step.

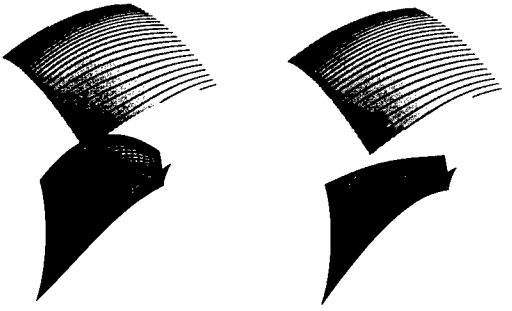
When a given boundary is a space curve, we have to approximate its evolute with third-degree Bézier curves for representing its torsion. In this paper, we only treat planar evolutes, but we can extend the method to the cases of space curves using the algorithm given in [7].

We show some examples of surfaces generated from four boundary curves with different types of curvature distribution. In Fig. 3, both boundary curves have monotonic curvature, but in Fig. 4, they have opposite curvature changes. On the other hand in Fig. 5 one boundary curve has a maximum curvature radius in the middle. Each figure shows an evolute surface and the generated surface with a silhouette pattern on the surface. Figures (a), (c) and (e) are the results of linear interpolation of the shape parameters, while the parameters are determined to get the minimum highlight line criterion in figures (b), (d) and (f). All the surfaces are smoothly generated, but from the point of



(c) (d)

Fig. 4. Example 2 of smooth highlight lines. (c) linear interpolation: $H = 11.24 \times 10^{-3}$, (d) minimum H: $H = 0.139 \times 10^{-3}$.



(e) (f)

Fig. 5. Example 3 of smooth highlight lines. (e) linear interpolation: $H = 0.450 \times 10^{-3}$, (f) minimum H: $H = 0.067 \times 10^{-3}$.

highlight-line smoothness, the surface quality is much improved in the right-hand figures. We cannot find the difference from usual surface evaluation, especially in Fig. 3, but H is reduced to less than one half. In Fig. 4 and Fig. 5, highlight lines are better and the shapes of the evolute surfaces are quite different.

§5. Surface Modification by Specified Highlight Line

Next, we consider modification of a surface according to the designer's intention. A designer wants to specify a highlight line on the surface by indicating the line to be changed. We determine the control variables in (6) so that the surface has the specified highlight line. We calculate the squared sum of the angle difference between tangent directions of the two highlight lines at

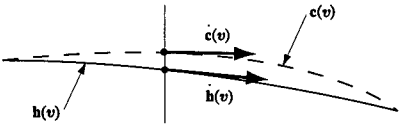


Fig. 6. Modification of surface by highlight line specification. A solid line $\mathbf{h}(v)$ is a highlight line to be modified and a dashed line $\mathbf{c}(v)$ is specified one.

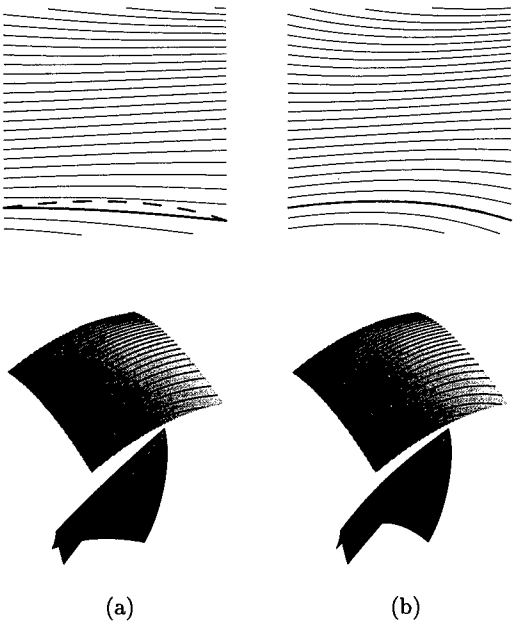


Fig. 7. Surface modification by highlight line specification.

the several corresponding points (see Fig. 6). Then, we change the surface to minimize the value of

$$E = \sum_{i=1}^{20} \arccos^2 \frac{\dot{\mathbf{c}}(v_i) \cdot \dot{\mathbf{h}}(v_i)}{|\dot{\mathbf{c}}(v_i)||\dot{\mathbf{h}}(v_i)|}. \tag{7}$$

Fig. 7 shows an example of surface modification. The upper two figures are projected highlight lines, and the lower two figures are an original surface and modified one with their evolute surfaces. In the left figure (a), a dashed line is a highlight line specified by a designer to change the corresponding highlight line (bold line). In the right figure (b), the surface is modified to have the specified highlight line. As a result, its silhouette pattern is changed, but the curvature distribution is smooth, as required by the designer.

§5. Summary

We have proposed generation and modification methods of surfaces which obtain not only smooth curvature distribution, but also smooth highlight lines. The generated surface is globally smooth because it is generated so that its evolute surface becomes smooth. Further, the shape of the evolute surface is determined to minimize the introduced highlight line criterion, while the surface satisfies the specified highlight line when it is given by a designer.

References

1. Beier, K-P. and Y. Chen, Highlight line algorithm for realtime surface-quality assessment, Comput. Aided Design **26**, 4 (1994), 268–277.
2. Chen, Y., K-P. Beier and D. Papageorgiou, Direct highlight line modification on nurbs surfaces, Comput. Aided Geom. Design **14** (1997), 581-601.
3. Farin, G., *Curves and Surfaces for CAGD 4th Edition*, Academic Press, San Diego, 1996.
4. Hagen, H. and G. P. Bonneau, Variational design of smooth rational Bezier curves, Comput. Aided Geom. Design **8** (1991), 393–399.
5. Higashi, M., I, Kohzen and J. Nagasaka, An interactive CAD system for construction of shapes with high-quality surface, in *Computer Application of Production and Engineering CAPE'83*, E. A. Warman (ed), North-Holland, Amsterdam, 1983, 371–390.
6. Higashi, M., T. Kushimoto and M. Hosaka, On formulation and display for visualizing features and evaluating quality of free-form surfaces, in *EUROGRAPHICS'90*, C. E. Vandodoni and D. A. Duce (eds), North-Holland, Amsterdam, 1990, 79–92.
7. Higashi, M., H. Tsutamori and M. Hosaka, Generation of surfaces by controlling curvature variation, Comput. Graphics Forum (EUROGRAPHICS'96) **15** 3 (1996), C-187–C-196.
8. Hoschek, J., and D. Lasser, translated by L. L. Schumaker, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Wellesley, 1993.
9. Klass, K., Correction of local surface irregularities using reflection lines, Comput. Aided Design **12**, 4 (1980), 73–76.
10. Meier, H. and H. Nowaki, Interpolating curves with gradual changes in curvature, Comput. Aided Geom. Design **4** (1987), 297–305.

Masatake Higashi, Hiroto Harada and Mitsuru Kuroda
Toyota Technological Institute
2-12-1, Hisakata, Tempaku-ku, Nagoya 468-8511, Japan
higahsi@toyota-ti.ac.jp, 920043@nhkspg.co.jp,
kuroda@toyota-ti.ac.jp

MIPS: An Efficient Global Parametrization Method

Kai Hormann and Günther Greiner

Abstract. The problem of parametrizing 3D data points is fundamental for many applications in computer-aided geometric design, e.g. surface fitting, texture mapping, and remeshing. We present a new method for constructing a global parametrization of a triangulated (topologically disk-like) surface over a planar region with minimal distortion. In contrast to many existing approaches which need the boundary of the parametrization to be fixed in advance, the boundary develops naturally with this new algorithm.

§1. Introduction

In general, a triangulated set of data points $P_i \in \mathbb{R}^3$ with triangles $T_j = \Delta(P_{j_0}, P_{j_1}, P_{j_2})$ and a 2-manifold domain $\Omega \subset \mathbb{R}^3$, over which the points are to be parametrized, are given. In most cases the domain is either planar ($\Omega \subset \mathbb{R}^2 \subset \mathbb{R}^3$) or a polygonal mesh with planar facets. The task is now to find parameter values $p_i \in \Omega$, one for each data point P_i, such that the topology of the point set is preserved, i.e., the triangles in the parameter domain $t_j = \Delta(p_{j_0}, p_{j_1}, p_{j_2})$ must not overlap.

After determining the parameter values, the interpolation problem can be written as follows: *find a function* $F : \Omega \to \mathbb{R}^3$ *with* $F(p_i) = P_i$ [5,8,12]. The simplest solution to this problem is the piecewise linear function that linearly maps each parameter triangle t_j to the corresponding surface triangle T_j (i.e., $F(t_j) = T_j$). This function is typically used in the case of texture mapping, where color information is defined in the parameter domain and mapped onto the 3D object to make it look more realistic [1,13]. The function F can also be used for remeshing the triangulated data points in order to get a mesh with regular connectivity, so that multiresolution analysis and subdivision techniques can be applied [3,10].

We only address the case of triangulated point sets that are topologically disk-like (i.e., having a boundary and no holes) and thus can be parametrized

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 153–162.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

over a simply connected planar domain $\Omega \subset \mathbb{R}^2$. In Section 2 a summary of the previous work dedicated to this problem is given and the limitations of the existing methods are outlined. Our method to overcome these limitations will be explained in detail in Section 3. In Section 4 we show the advantages of the presented approach, giving some examples of surface approximation and remeshing with the new parametrizations. The paper concludes with a discussion of the drawbacks of the proposed technique, and suggestions for future investigations.

§2. Previous Work

While it is quite clear how to solve the *local* problem, i.e., parametrizing a set of points surrounding a reference point R, which can be done e.g. by an exponential mapping or by projection into an adequate tangent plane at R, the *global* problem is more complicated and has been addressed in several earlier papers.

Bennis et al. [1] propose a method based on differential geometry: they map isoparametric curves of the surface onto curves in the parameter domain such that the geodesic curvature at each point is preserved. The parametrization is then extended to both sides of that initial curve until some distortion threshold is reached. But this method as well as the one presented in [13] by Maillot et al. require the surface to be split into several independent regions, and therefore cannot be seen as a solution to the global problem.

Ma and Kruth [12] project the data points P_i onto a parametric base surface $S : \Omega \to \mathbb{R}^3$, and the parameter values of the projected points are taken as p_i. The approaches in [3,5,8,14] have the following strategy in common:

1) find a parametrization for the *boundary points*,
2) minimize an edge-based energy function

$$E = \frac{1}{2} \sum_{\{i,j\} \in \text{Edges}} c_{ij} \, \|p_i - p_j\|^2 \tag{1}$$

to determine the parametrization for the *inner points*.

The edge coefficients c_{ij} can be chosen in different ways. While Floater chooses them so that the geometric shape of the surface is preserved [5], Greiner and Hormann set $c_{ij} = \frac{1}{\|P_i - P_j\|^r}$ for some $r \geq 0$, as they want to minimize the energy of a network of springs [8]. Both methods are generalizations of well-known results for the parametrization of curves [4,6,11]. Furthermore, Taubin used the energy function (1) for smoothing polyhedral surfaces [14] and found $r = 1$ to produce good results.

A different method is introduced by Pinkall and Polthier in [14], and by Eck et al. in [3], where the *Dirichlet energy* of the piecewise linear function F^{-1} that maps the surface triangles T_j to the corresponding parameter triangles t_j is considered. It can be expressed as in (1) with $c_{ij} = \frac{1}{4}(\cot\alpha + \cot\beta)$, where α and β are the angles opposite to the edge $\overline{P_i P_j}$ in the two adjacent surface triangles.

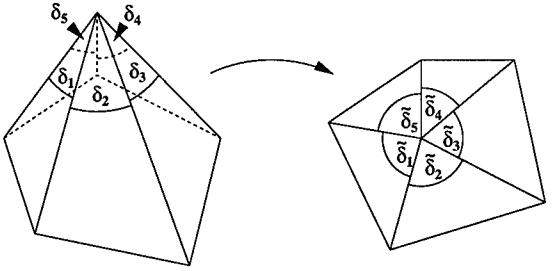


Fig. 1. A pyramid cannot be parametrized without distortions.

In all cases, minimizing (1) is equivalent to solving a non-singular sparse positive definite matrix system, that is (apart from Floater's method) even symmetric. Though this is a comparatively fast way to find a parametrization, it suffers from the fact that it is not clear how to choose the initial parametrization of the boundary points. Floater maps them to the boundary of the unit square using chord length parametrization, Greiner and Hormann project them into the plane that fits all boundary points best in the least square sense, and Eck et al. use parameter values lying on a circle. Note the importance of choosing a convex configuration for the boundary points, since triangle flipping may occur otherwise. Triangle flipping can also be caused by negative weights c_{ij}, which may happen with the method of [3,14] at sharp peaks.

These techniques seem to be rather arbitrary and do not take the geometry of the boundary points into account. In the next section we will introduce a parametrization method that yields parameter values not only for the inner points, but also for the boundary points. Since this method also generates parametrizations that are "as isometric as possible" (i.e., having minimal distortion), we will call them: **M**ost **I**sometric **P**arametrization**S** (MIPS).

§3. MIPS—Most Isometric Parametrizations

Let us briefly review the situation: we are given a set of triangulated data points $P_i \in \mathbb{R}^3$ with a boundary and no holes, and want to find a parametrization, i.e., a set of parameter values $p_i \in \mathbb{R}^2$ so that the topology is preserved. In order to define the quality of a parametrization, we consider the piecewise linear interpolation function $f : \mathbb{R}^3 \to \mathbb{R}^2$ that maps the data points to the corresponding parameter values, i.e., $f(P_i) = p_i$.

As the triangulated surface may be geometrically complex, this function will inevitably cause some deformation to the shape of the triangles. Consider e.g. the configuration in Fig. 1, which can only be parametrized without any deformations if the angles δ_i add up to 2π. In general, only for developable surfaces (e.g. planes, cylindrical and conical surfaces) an isometric parametrization without any distortion can be found. To keep the distortion as small as possible, we must somehow measure this deformation so that the best parametrization can be found in a minimization process.

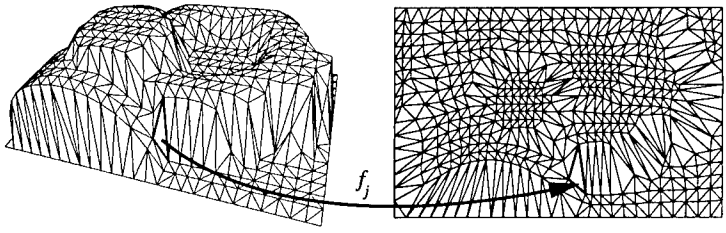


Fig. 2. An atomic linear map between surface and parameter triangle.

Clearly, f can be decomposed into *atomic linear maps* f_j (see Fig. 2) that map a surface triangle $T_j = \Delta(P_{j_0}, P_{j_1}, P_{j_2})$ to the corresponding parameter triangle $t_j = \Delta(p_{j_0}, p_{j_1}, p_{j_2})$. Thus it is sufficient to measure the distortion of linear maps: if E were such a deformation functional, the best parametrization could simply be found by minimizing $\sum_j E(f_j)$.

While Pinkall and Polthier in [14] and Eck et al. in [3] consider the Dirichlet energy $E_D(f) = \frac{1}{2}\int \|\nabla f\|^2$ as a measure of deformation, Maillot et al. propose the Green-Lagrange deformation tensor $\|I_f - \mathrm{Id}\|^2$ that describes the distance of the first fundamental form of f, $I_f = \nabla f^t \cdot \nabla f$, to the identity matrix in some 2×2-matrix norm [13].

An energy functional that measures the deformation of a linear function should have the following properties: it should be

1) unaffected by translations,
2) unaffected by orthogonal transformations,
3) unaffected by scalings,

since the shape of triangles is not changed by these operations. Furthermore, it is desirable to avoid degeneracies, so we need

4) a functional that punishes collapsing triangles very badly.

Notice that the Dirichlet energy meets the first and second condition but favors small parameter triangles, contradicting the other two conditions. Indeed, if the parameter values of the boundary points are not fixed, the minimum of that functional is the singular parametrization where all parameter values p_i collapse to one point. The Green-Lagrange deformation tensor also fails to meet the third and fourth condition and the second one is only fulfilled if the chosen matrix norm is invariant to orthogonal transformations.

Now, let $g(x) = Ax + b$ be an atomic linear map that maps a surface triangle T to a parameter triangle t. Note that by introducing a local coordinate system at T, this function can be seen as a mapping from $\mathbb{R}^2 \to \mathbb{R}^2$.

Because of the first condition, the constant part b of the function g should not be taken into account by the desired deformation functional. Remembering the singular value decomposition of a matrix $U^t A V = \Sigma = \begin{pmatrix} \sigma_1 & \\ & \sigma_2 \end{pmatrix}$, where $\sigma_1 \geq \sigma_2$ are the singular values of A and U and V are orthogonal matrices, the functional should further depend only on σ_1 and σ_2, thus fulfilling the second condition. As the singular values are the lengths of the semi-axes of

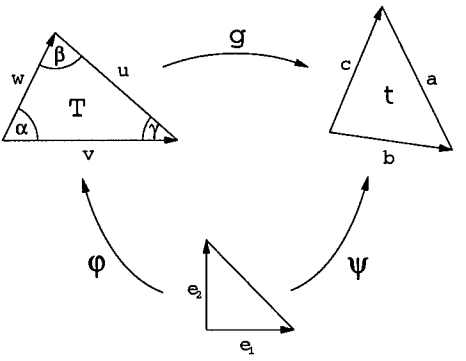


Fig. 3. Decomposition of a linear map g.

the ellipse $\{Ax : \|x\|_2 = 1\}$, the ratio σ_1/σ_2 seems to be a good measure of the deformation of g that also fulfills the third and fourth condition by punishing vanishing triangles with ∞. From Linear Algebra it is known [7] that this ratio is the 2-norm condition number of the matrix:

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sigma_1}{\sigma_2}.$$

Since the 2-norm condition number of even a 2×2-matrix is rather costly in numerical computations, we decided to use the condition number based on the *Frobenius Norm* $\|\cdot\|_F$ instead, which still meets the four conditions and is much easier to handle:

$$\begin{aligned}\kappa_F(A) = \|A\|_F\|A^{-1}\|_F &= \sqrt{\sigma_1^2+\sigma_2^2}\sqrt{(\tfrac{1}{\sigma_1})^2+(\tfrac{1}{\sigma_2})^2} \\ &= \frac{\sigma_1^2+\sigma_2^2}{\sigma_1\sigma_2} = \frac{\sigma_1}{\sigma_2}+\frac{\sigma_2}{\sigma_1} = \kappa_2(A)+\frac{1}{\kappa_2(A)} \qquad (2) \\ &= \frac{\operatorname{trace}(A^tA)}{\det A}. \qquad (3)\end{aligned}$$

From (2) we can see how close κ_F and κ_2 are related and that it is no major difference whether we minimize the one or the other in order to get linear mappings with low distortion. We will now use (3) to get a representation of κ_F that is suitable for numerical computations.

If we decompose the linear function g according to Fig. 3, where $\{e_1, e_2\}$ is the canonical basis in $\mathbb{R}^2$, we have $g = \psi \circ \varphi^{-1}$. Further, we have $A = \partial g = \partial\psi\partial\varphi^{-1}$ and a little calculation (see [14]) yields

$$\begin{aligned}\kappa_F(g) := \kappa_F(A) = \frac{\operatorname{trace}(A^tA)}{\det A} &= \frac{\operatorname{trace}(\partial g^t\partial g)}{\det\partial\psi\det\partial\varphi^{-1}} = \frac{4E_D(g)}{\det\partial\psi} \qquad (4) \\ &= \frac{\cot\alpha|a|^2+\cot\beta|b|^2+\cot\gamma|c|^2}{\det\partial\psi}.\end{aligned}$$

From (4) and the observation that $\det\partial\psi$ is twice the area of the parameter triangle t, we can interpret the deformation energy κ_F as the Dirichlet energy per parameter area.

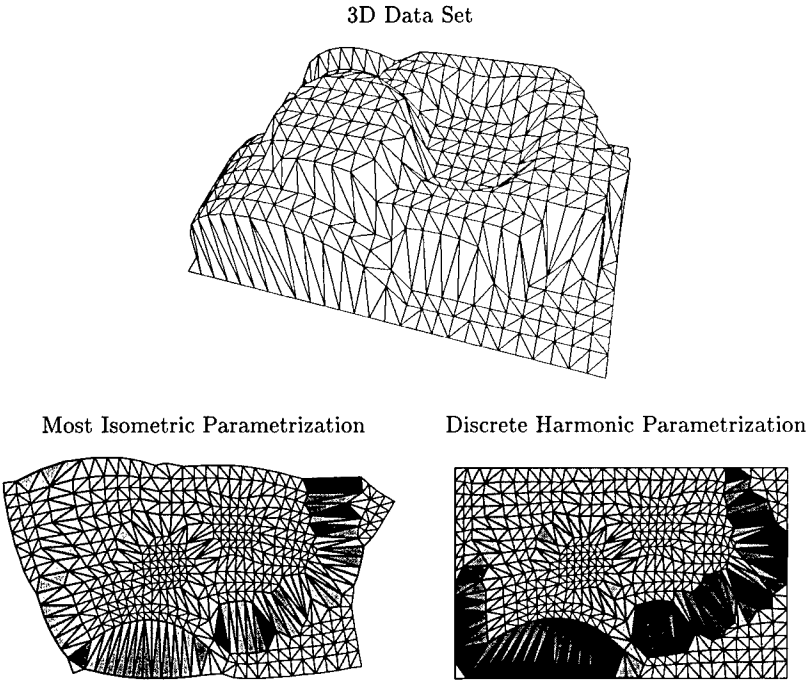


Fig. 4. Data set and gray-coded Dirichlet energy of different parametrizations.

Now, by minimizing the deformation functional $\kappa = \sum_j \kappa_F(f_j)$, we will get a set of parameter values $p_i \in \mathbb{R}^2$ that defines a parametrization with minimal distortion. Note that, as κ meets the four conditions from above, the minimum will only be unique up to movements and scalings. Anyway, this is not a drawback and can be fixed by retaining two arbitrarily chosen parameter values.

The main advantage of the proposed approach is that it is no longer necessary to fix the parameter values of the boundary points in advance. Instead, the boundary of the parametrization will develop most naturally in such a way that the deformation energy κ is minimized.

§4. Examples

We now illustrate the advantages of our new approach by showing some examples of surface approximation and texture mapping with different parametrizations.

In Fig. 4, a triangulated surface with 476 data points and 864 triangles can be seen. This data set has been parametrized with our new method and by minimizing the Dirichlet energy according to [3,14] which produced the best results of all the approaches mentioned in Sec. 2. Since the minimization of the Dirichlet energy can also be interpreted as a discrete harmonic mapping (see [3]), we will call the result Discrete Harmonic Parametrization.

Most Isometric Parametrization Discrete Harmonic Parametrization

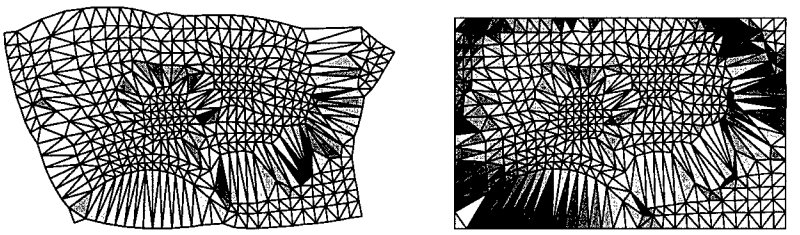


Fig. 5. Gray-coded κ deformation energy of different parametrizations.

We have encoded the amount of Dirichlet energy per triangle as gray tones (white color signifies low and black color denotes high energy). One can clearly see that the second method generates great deformations especially near the border of the parametrization which is due to the arbitrarily chosen parameter values for the boundary points. This effect is even more distinct if we look at the κ deformation energy per triangle, which has been gray-coded in Fig. 5.

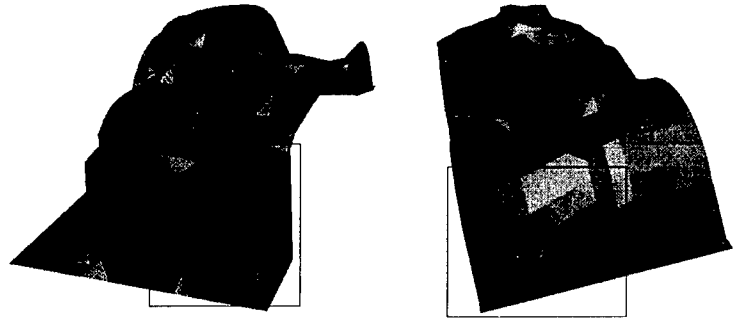
Fig. 6 shows how the deformations of these parametrizations affect an approximating surface. We have gray-coded the *mean curvature* of the surfaces in order to emphasize the fact that strong deformations in the parametrization of the data points cause the surface to wrinkle in these areas (dark color refers to high absolute values of the mean curvature).

Finally, Fig. 7 shows an example of remeshing. A base mesh of 4 triangles has been split six times by a regular 1-to-4-split, generating a triangle mesh with 16,384 triangles and regular connectivity, i.e., all vertices have valence 6 except for the one that refers to the central vertex of the base mesh and the boundary vertices. The Hoppe mannequin head has been remeshed with this semi-regular mesh using different parametrizations. The *Chord Length Parametrization*, used in the example to the right, refers to (1) with $c_{ij} = \frac{1}{\|P_i - P_j\|}$ (see [8] for details).

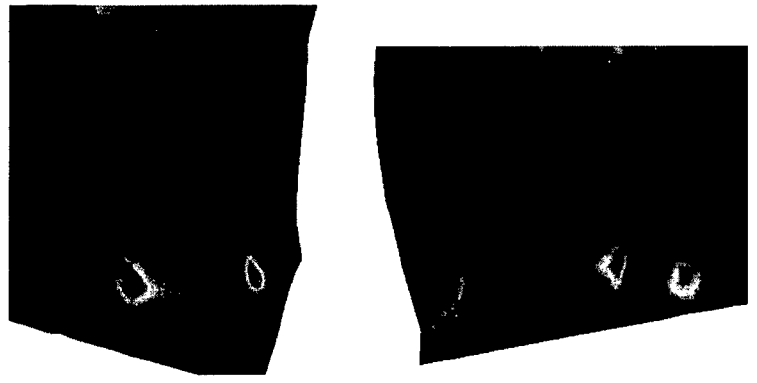
§5. Conclusion

We have presented a new method for constructing parametrizations of triangulated surfaces with a boundary. This parametrization can be used for surface approximation, texture mapping, and remeshing of the original mesh. The main advantage of our approach is that, in contrast to existing methods, the parametrization of the boundary data points is done in the same way as the parametrization of the inner points. Therefore we think that our approach is more natural than the other methods which set the parameter values at the boundary heuristically.

Often, the problem of parametrizing triangulated surfaces with holes occurs. So far, we have tacitly ignored this problem, but the proposed method is capable of dealing with these situations, in principle. Nevertheless, it might happen that overlapping parameter triangles will be generated at holes, which



Most Isometric Parametrization



Discrete Harmonic Parametrization

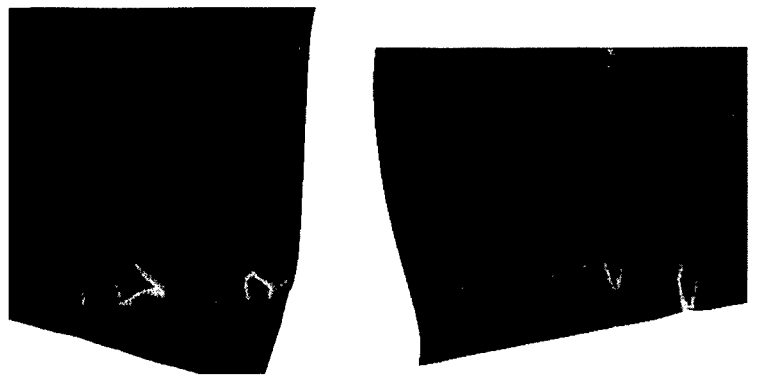


Fig. 6. Curvature plot of approximating surfaces with different parametrizations.

Most Isometric Parametrization Chord Length Parametrization

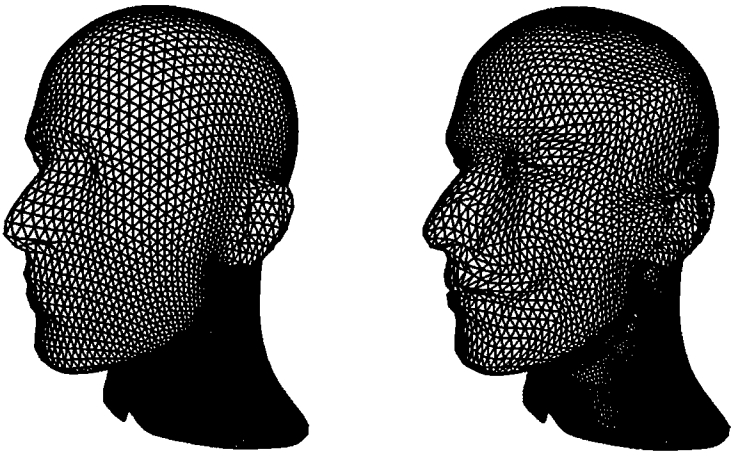


Fig. 7. Remeshing with different parametrizations.

can be fixed by triangulating the hole in a preprocessing step and removing the additional triangles afterwards.

The main drawback of our approach is that it requires the minimization of a rational quadratic function, while the other parametrization techniques only need to minimize a quadratic term which can be done by solving a sparse linear system of equations.

Our future work will therefore be concentrated on developing hierarchical methods for efficiently solving the problem. The concept of progressive meshes, introduced by Hoppe in [9] seems to be a good basis for such investigations.

Acknowledgments. This work was partly supported by the Deutsche Forschungsgemeinschaft through the Sonderforschungsbereich 603 Modellbasierte Analyse und Visualisierung komplexer Sensordaten.

References

1. Bennis, C., J.-M. Vézien, and G. Iglésias, Piecewise surface flattening for non-distorted texture mapping, Computer Graphics (SIGGRAPH '91 Proceedings) **25** (1991), 237–246.
2. Campagna, S., and H.-P. Seidel, Parameterizing meshes with arbitrary topology, in *Image and Multidimensional Digital Signal Processing '98*, H. Niemann, H.-P. Seidel, and B. Girod (eds.), infix, 1998, 287–290.
3. Eck, M., T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, Multiresolution analysis of arbitrary meshes, Computer Graphics (SIGGRAPH '95 Proceedings) **29** (1995), 173–182.

4. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design*, Acadamic Press, Boston, 1993.
5. Floater, M. S., Parameterization and smooth approximation of surface triangulations, Comput. Aided Geom. Design **14** (1997), 231–250.
6. Foley, T. A., and G. M. Nielson, Knot selection for parametric spline interpolation, in *Mathematical Methods in Computer Aided Geometric Design*, T. Lyche and L. L. Schumaker (eds.), Academic Press, New York, 1989, 261–271.
7. Golub, G. H., and C. F. van Loan, *Matrix Computations*, The John Hopkins University Press, Baltimore, Maryland, 1989.
8. Greiner, G., and K. Hormann, Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines, in *Surface Fitting and Multiresolution Methods*, A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1997, 163–172.
9. Hoppe, H., Progressive meshes, Computer Graphics (SIGGRAPH '96 Proceedings) **30** (1996), 99–108.
10. Lee, A. W. F., W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, MAPS: Multiresolution adaptive parameterization of surfaces, Computer Graphics (SIGGRAPH '98 Proceedings) **32** (1998), 95–104.
11. Lee, E. T. Y., Choosing nodes in parametric curve interpolation, Computer-Aided Design **21** (1989), 363–370.
12. Ma, W., and J. P. Kruth, Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces, Computer-Aided Design **27** (1995), 663–675.
13. Maillot, J., H. Yahia, and A. Verroust, Interactive texture mapping, Computer Graphics (SIGGRAPH '93 Proceedings) **27** (1993), 27–34.
14. Pinkall, U., and K. Polthier, Computing discrete minimal surfaces and their conjugates, Experimental Mathematics **2** (1993), 15–36.
15. Taubin, G., A signal processing approach to fair surface design, Computer Graphics (SIGGRAPH '95 Proceedings) **29** (1995), 351–358.

Kai Hormann
Computer Graphics Group
University of Erlangen-Nürnberg
Am Weichselgarten 9
D-91058 Erlangen, Germany
hormann@informatik.uni-erlangen.de

Günther Greiner
Computer Graphics Group
University of Erlangen-Nürnberg
Am Weichselgarten 9
D-91058 Erlangen, Germany
greiner@informatik.uni-erlangen.de

Comparison of Different Multisided Patches Using Algebraic Geometry

Kęstutis Karčiauskas and Rimvydas Krasauskas

Abstract. Different constructions of multisided surface patches (due to Sabin, Hosaka-Kimura, Warren, Loop-DeRose, etc.) are studied via considering base points of their parametrizations. This analysis shows hidden interrelations between various cases and enables to find new efficient control point schemes in more general situations. In particular, toric patches are introduced.

§1. Introduction

The problem of smooth filling of m-sided holes arises in many modeling situations. It is solved using various methods: recursive subdivision, surface splitting, data blending and control point schemes. We consider here only the case when a m-sided patch is defined via control points as a single piece bounded by Bézier curves of degree n. M. Sabin [11] introduced 3- and 5-sided patches bounded by conics ($n = 2$) and suitable for an inclusion in B-spline surface. Hosaka and Kimura [2] proposed the same type of patches with $n = 3$. Zheng and Ball [15] extended the previous constructions to arbitrary degree n. In the same fashion, 6-sided patches were constructed (see [2,12,15]). Unfortunately, these 6-sided patches seems to be nonrational. Loop and DeRose [9] introduced rational S-patches, and used them in [10] for building Sabin and Hosaka–Kimura-like patches ($n = 2, 3$) with arbitrary number of sides m. As far as we know, Warren was the first to introduce the method of blowing up base points (well-known in algebraic geometry) to the CAGD community. He used it in [14] for creating 5-, 6-sided patches. Analysis of mentioned approaches and the convex combination method (cf. Gregory [1]) shows that m-sided patches for $m > 4$ should be rational. Hence it is natural to use theoretical results from algebraic geometry concerning rational surfaces. The method of base points enabled Karčiauskas [3] to build well structured rational 5-sided patch with actually the same properties as the original Warren hexagon. In [4] these patches are used for creating 5- and 6-sided Sabin and

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 163–172.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

Hosaka–Kimura-like surface patches with boundary curves of arbitrary degree n. Similar patches over a regular m-gon for any m (except 4) and for arbitrary n are obtained in [5] also using the base point method. The patches in [4, 5] have lower degree parametrization than previous ones. We call them T-patches. Moreover, the base points method is good for building bridges between various approaches, especially in pentagonal case. In 6-sided case the relations are more complicated. On the other hand, it appeared that this hexagonal patch belongs to a special class of so-called toric surfaces, which were studied in detail in algebraic geometry. First applications of toric varieties in CAGD were demonstrated by Warren [14] and Krasauskas [7].

In this paper we describe initially hidden interrelations between pentagonal Sabin, Hosaka–Kimura and Loop–DeRose patches via the T-patch concept. Six-sided patches are considered using both base points and toric methods. Five- and six-sided cases are actually most important (beside triangular and rectangular patches) in geometric modeling and at the same time most convenient from the algebraic geometry point of view. Here we only outline results. Full proofs can be found in papers [4, 5, 8] of the authors. Relations between triangular Sabin, Hosaka–Kimura and Loop–DeRose patches are described in [5]. Algebraic version of convex combination patches is presented in [6].

§2. Notations and Definitions

In order to consider several variants of multisided patches defined via control points, we recall the most general concept of a rational patch.

Definition 1. *A* rational surface patch *is a mapping* $F : D \to \mathbb{R}^k$ *defined on a domain* $D \subset \mathbb{R}^2$ *by the formula*

$$F(\boldsymbol{t}) = \frac{\sum_{q\in\mathcal{I}} w_q \boldsymbol{p}_q f_q(\boldsymbol{t})}{\sum_{q\in\mathcal{I}} w_q f_q(\boldsymbol{t})}, \tag{1}$$

where polynomial functions f_q *labeled by some set* $\mathcal{I}$ *are called* basis functions, *the points* $\boldsymbol{p}_q \in \mathbb{R}^k$ *are* control points, *and the numbers* w_q *are their* weights.

The Sabin and Hosaka–Kimura-like patches (see [2,4,5,10,11,15]) behave like tensor product surfaces along their boundaries, and can be connected smoothly with surrounding rectangular patches. We denote a patch of this type by SHK_m^n, where m is a number of boundary curves and n is their degree.

Let $\boldsymbol{w}_0, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m-1}$ be the vertices of a regular m-gon with a center $\boldsymbol{w}$ and let n be a fixed natural number. For each triangle with the vertices $\boldsymbol{w}, \boldsymbol{w}_s, \boldsymbol{w}_{s+1}$, $0 \le s \le m-1$, the points

$$\boldsymbol{w}_{ij}^s = \frac{i}{n}\boldsymbol{w} + \frac{j}{n}\boldsymbol{w}_{s+1} + \frac{n-i-j}{n}\boldsymbol{w}_s, \quad i, j \ge 0,\ i+j \le n, \tag{2}$$

linked together form a triangulation of an m-gon (see Fig. 1). The set of all its vertices is denoted by $\mathcal{L}_m^n$. It is convenient to enumerate them by the triples

$$(s,i,j), \quad 0 \le s \le m-1,\ 0 \le i \le n,\ 0 \le j \le n-i,$$

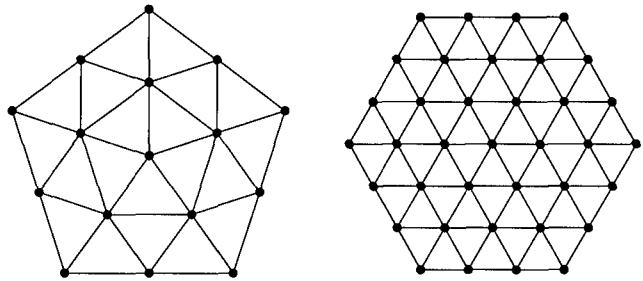


Fig. 1. Control point schemes of T-patches.

where triples $(s, i, n-i)$ and $(s+1, i, 0)$ are identified (the first index s is treated in a cyclic fashion). Indices s, i, j correspond to labeling in the formula (2). The graphs $\mathcal{L}_m^n$ define a combinatorial structure on the control point nets of T-patches.

The domain of some patches is a regular m-gon. In this case we assume linear functions have inward-oriented normal vectors. For $0 \le s \le m-1$, we write $\hat{l}_s$ for the function defining a line $\overline{\boldsymbol{w}_s \boldsymbol{w}_{s+1}}$. An intersection of the lines $\overline{\boldsymbol{w}_{s-1} \boldsymbol{w}_s}$ and $\overline{\boldsymbol{w}_{s+1} \boldsymbol{w}_{s+2}}$ is denoted by $\boldsymbol{b}_s$. By $\bar{l}_s$ we denote a function defining a line $\overline{\boldsymbol{b}_{s-1} \boldsymbol{b}_s}$.

Using the blowing up method (see [3,13]) a 5-sided patch is defined via basis functions vanishing simultaneously at the two vertices $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ of the domain triangle $\Delta \boldsymbol{v}_0 \boldsymbol{v}_1 \boldsymbol{v}_2$. A 6-sided patch is defined via basis functions vanishing simultaneously at all three vertices. In these cases we denote by l_0, l_1, l_2 the barycentric coordinates of a point with respect to the triple $\boldsymbol{v}_0$, $\boldsymbol{v}_1$, $\boldsymbol{v}_2$. The infinite points corresponding to the lines $\overline{\boldsymbol{v}_0 \boldsymbol{v}_1}$ and $\overline{\boldsymbol{v}_0 \boldsymbol{v}_2}$ are denoted by $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ respectively.

Definition 2. *A function f has a zero of multiplicity μ at a point $\boldsymbol{p}$ if it vanishes at $\boldsymbol{p}$ together with all partial derivatives up to the order $\mu - 1$. A point $\boldsymbol{p}$ is a* base point *of multiplicity μ of a rational map (1) if all basis functions f_q have a zero of multiplicity μ at $\boldsymbol{p}$.*

For a set of planar points $\mathcal{X} = \{\boldsymbol{p}_0, \ldots, \boldsymbol{p}_s\}$, we denote by $\mathcal{P}(k, \mu, \mathcal{X})$ the linear space of polynomials of degree k which have zero of multiplicity μ at all points $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_s$.

§3. T-patches

Defining 5- and 6-sided T-patches, we set $\mathcal{I} = \mathcal{L}_5^n$ and $\mathcal{I} = \mathcal{L}_6^n$ respectively. Various type of basis functions for 5- and 6-sided patches are defined using the following scheme. Assume there are $m+1$ functions $h_0, h_1, \ldots, h_{m-1}, h$ $(m = 5, 6)$ and positive numbers k_{ij}^n, $0 \le i \le n$, $0 \le j \le n-i$, satisfying the symmetry conditions $k_{ij}^n = k_{i,n-i-j}^n$. For $q = (s, i, j) \in \mathcal{L}_m^n$, the functions f_q

are defined by the formula

$$f_q = k_{ij}^n h_s^{n-i-j} h_{s+1}^j h^i. \tag{3}$$

Now we specify the functions h_s, h.

Definition 3. *Five-sided T_5^n-patch and six-sided T_6^n-patch are defined over a triangle via the formulas*

$$T_5^n : \begin{cases} h_0 = l_0^2, \quad h_1 = l_0 l_1 (l_0 + l_1), \quad h_2 = l_1^2 l_2, \quad h_3 = l_1 l_2^2, \\ h_4 = l_0 l_2 (l_0 + l_2), \quad h = l_0 l_1 l_2, \end{cases} \tag{4}$$

$$T_6^n : \begin{cases} h_0 = l_0^2 l_1, \quad h_1 = l_0 l_1^2, \quad h_2 = l_1^2 l_2, \quad h_3 = l_1 l_2^2, \\ h_4 = l_0 l_2^2, \quad h_5 = l_0^2 l_2, \quad h = l_0 l_1 l_2. \end{cases}$$

A five-sided $\widetilde{T}_5^n$-patch and six-sided $\widetilde{T}_6^n$-patch are defined over a regular pentagon and hexagon, respectively, via

$$\widetilde{T}_5^n : h_s = \hat{l}_{s+1} \hat{l}_{s+2}^2 \hat{l}_{s+3} \bar{l}_s, \quad s = 0, 1, \ldots, 4, \quad h = \prod_{s=0}^{4} \hat{l}_s, \tag{5}$$

$$\widetilde{T}_6^n : h_s = \hat{l}_{s+1} \hat{l}_{s+2}^2 \hat{l}_{s+3}^2 \hat{l}_{s+4}, \quad s = 0, 1, \ldots, 5, \quad h = \prod_{s=0}^{5} \hat{l}_s.$$

If $k_{0j}^n = \binom{n}{j}$, the boundary curves are Bézier curves of degree n. So the boundary curves are integral if their weights are equal to 1, though the patches are rational for any choice of the other weights.

From the designers point of view, it is convenient when a cyclic change of the input data does not change a patch as an image in $\mathbb{R}^3$. The $\widetilde{T}_5^n$- and $\widetilde{T}_6^n$-patches are symmetric by definition. The T_5^n- and T_6^n-patches are also symmetric (see [4, 5]). Their cyclic reparametrizations are given by the birational transformations of the domain triangle (Cremona transformations) of order 5 and 6 respectively.

Remark 4. *It is shown in [4] that T_5^n- and T_6^n-patches give the same class of the surfaces as $\widetilde{T}_5^n$- and $\widetilde{T}_6^n$-patches. So we actually have two kinds of parametrizations of 5- and 6-sided surfaces. The $\widetilde{T}_5^n$- and $\widetilde{T}_6^n$-patches can be easier handled using standard methods, since they are defined over traditional symmetric domain. The T_5^n- and T_6^n-patches are more convenient from the algebraic geometry point of view. For example, the latter approach gives the third type of parametrization of T-patches, which is suitable for an efficient plotting: T_5^n-patch can be represented as a collection of three Bézier patches of bidegree $(2n, 2n)$; T_6^n-patch can be represented as a collection of six Bézier patches of the same bidegree.*

The principles of blowing up and plotting T_5^n-patches are shown in Fig. 2.

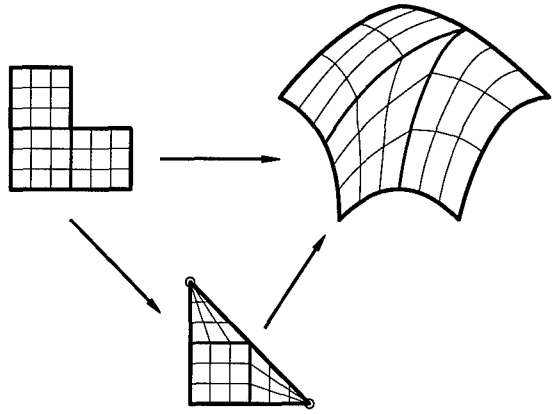


Fig. 2. Blowing up and plotting T_5^n-patches.

Lemma 5. *The basis functions of all T-patches are linearly independent. Moreover, the spaces* $\mathcal{P}(3n, n, \{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{e}_1, \boldsymbol{e}_2\})$, $\mathcal{P}(3n, n, \{\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2\})$, $\mathcal{P}(5n, 2n, \{\boldsymbol{b}_s, s = 0, \ldots, 4\})$ *are generated by the basis functions of the* T_5^n-, T_6^n- *and* $\widetilde{T}_5^n$*-patches respectively.*

Lemma 5 enables us to establish algebraic relations between different surfaces.

§4. Interrelations Between Pentagonal Patches

We denote by U_5 a surface in $\mathbb{R}^5$ defined via equations $x_s - 1 + x_{s+2}x_{s+3} = 0$, $s = 1, \ldots, 5$ (the index s is treated in a cyclic fashion). This surface was introduced by Sabin [11]. A domain D for the pentagonal patches from [2,11,15] is a region in U_5 with $x_s \geq 0$, $s = 1, \ldots, 5$.

The interrelation mappings are defined via formula (1), assuming that $\mathcal{I} = \mathcal{L}_5^1$, $k_{00}^1 = 1$ and all weights are equal to 1. We set for simplicity $\boldsymbol{p}_s = \boldsymbol{p}_{s00}$, $\boldsymbol{p} = \boldsymbol{p}_{000}$, and denote $\boldsymbol{r}_0 = (0, 1, 1, 1, 0)$, $\boldsymbol{r}_1 = (0, 0, 1, 1, 1)$,..., $\boldsymbol{r}_4 = (1, 1, 1, 0, 0)$, $\boldsymbol{r} = (2/3, \ldots, 2/3)$ ($\boldsymbol{r}_s$ are the corner points of the Sabin domain). By $\boldsymbol{c}$ is denoted a barycenter of the triangle $\Delta \boldsymbol{v}_0 \boldsymbol{v}_1 \boldsymbol{v}_2$.

Definition 6. *Define rational mappings* $H_5, \widetilde{H}_5 : \mathbb{R}^2 \to \mathbb{R}^2$ *and* $G_5, \widetilde{G}_5 : \mathbb{R}^2 \to \mathbb{R}^5$ *as follows.* H_5 *and* G_5 *are defined fixing basis functions (4) with* $k_{10}^1 = 5(\sqrt{5} - 1)/2$ *and taking control points* $\boldsymbol{p}_s = \boldsymbol{w}_s$, $\boldsymbol{p} = \boldsymbol{w}$ *and* $\boldsymbol{p}_s = \boldsymbol{r}_s$, $\boldsymbol{p} = \boldsymbol{r}$ *respectively.* $\widetilde{H}_5$ *and* $\widetilde{G}_5$ *are defined fixing functions (5) with* $k_{10}^1 = 3(\sqrt{5} + 1)/2$ *and taking control points* $\boldsymbol{p}_0 = \boldsymbol{v}_0$, $\boldsymbol{p}_1 = \boldsymbol{v}_1$, $\boldsymbol{p}_2 = \boldsymbol{v}_1$, $\boldsymbol{p}_3 = \boldsymbol{v}_2$, $\boldsymbol{p}_4 = \boldsymbol{v}_2$, $\boldsymbol{p} = \boldsymbol{c}$ *and* $\boldsymbol{p}_s = \boldsymbol{r}_s$, $\boldsymbol{p} = \boldsymbol{r}$ *respectively.*

Theorem 7. *The mappings* G_5 *and* $\widetilde{G}_5$ *define parametrizations of the surface* U_5. *They map triangular and regular pentagonal domains respectively onto the Sabin domain. Moreover,* $G_5 \circ \widetilde{H}_5 = \widetilde{G}_5$, $\widetilde{H}_5 \circ H_5 = \mathrm{id}$, $H_5 \circ \widetilde{H}_5 = \mathrm{id}$.

Corollary 8. *Five-sided Sabin [11] and Hosaka-Kimura [2] patches can be represented as T_5^3- and T_5^4-patches respectively.*

Proof: The basis functions of the Sabin and Hosaka–Kimura patches are special polynomials of degree 12 and 20 respectively, which sum to 1 on U_5. Calculations (with MAPLE) give that their compositions with G_5 have the form Ag_r and Bg'_p respectively, where A, B are some rational functions, $g_r \in \mathcal{P}(9,3,\{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{e}_1,\boldsymbol{e}_2\})$, $g'_p \in \mathcal{P}(12,4,\{\boldsymbol{v}_1,\boldsymbol{v}_2,\boldsymbol{e}_1,\boldsymbol{e}_2\})$. Now the proof follows from Lemma 5. □

Notice, SHK_5^n-patches in [4] can be represented as T_5^n-patches. Let $\mathcal{I} = \{1,2,3,4,5\}$, $f_s = \hat{l}_s\hat{l}_{s+1}\hat{l}_{s+2}$, $s \in \mathcal{I}$, $\boldsymbol{p}_1 = (1,0,0,0,0),\ldots,\ \boldsymbol{p}_5 = (0,0,0,0,1)$. If all weights are equal to 1, the formula (1) defines a map $L : \mathbb{R}^2 \to \mathbb{R}^5$. An image of the map L is denoted by U'_5. The surface U'_5 is used in [9] for a definition of 5-sided S-patches. A domain of 5-sided S-patch is a regular pentagon.

Proposition 9. *A five-sided S-patch of depth n over regular pentagon can be represented as $\widetilde{T}_5^n$-patch.*

Proof: The basis functions of an S-patch of depth n (see [9]) are the compositions of the map L with the homogeneous polynomials of degree n. They are polynomials in $\mathcal{P}(3n,n,\{\boldsymbol{b}_s, s=0,\ldots,4\})$. Multiplication of the basis functions by C^n, where $C=0$ defines a circle going through the points $\boldsymbol{b}_s$, does not change the patch. New polynomials are in $\mathcal{P}(5n,2n,\{\boldsymbol{b}_s, s=0,\ldots,4\})$. Hence the original S-patch can be represented as $\widetilde{T}_5^n$-patch. □

We have seen, that Sabin and Hosaka–Kimura patches can be considered as the patches over a regular domain or over the Sabin domain in U_5. Similarly, an S-patch can be considered over the domain in U'_5 with nonnegative coordinates. We call it a Loop–DeRose domain.

Proposition 10. *There exists a mapping $p : U_5 \to U'_5$, which maps the Sabin domain onto the Loop–DeRose domain and $L = p \circ \widetilde{G}_5$.*

Proof: Define p as a composition of the projective transformation

$$y_i = x_i + x_{i+2} + x_{i+4} - a(x_{i+1} + x_{i+3}) + a - 2 \quad i = 1,\ldots,5,$$
$$y_0 = (3-2a)(x_1 + x_2 + x_3 + x_4 + x_5 + 2a).$$

$(a = (\sqrt{5}+1)/2)$ with the projection from a point on U_5: $p : (y_0,\ldots,y_5) \mapsto (y_1/y_0,\ldots,y_5/y_0)$. □

From the algebraic geometry point of view, the surface U_5 is more universal in the algebraic constructions than U'_5. As a confirmation of this property, we have that SHK_5^2- and SHK_5^3-patches in [10] can be represented only as T_5^5- and T_5^6-patches, respectively.

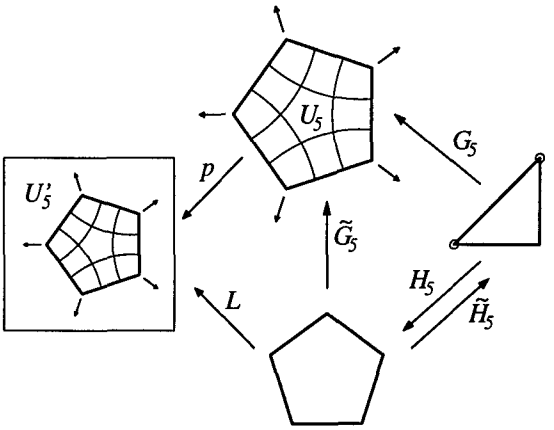


Fig. 3. Interrelations of 5-sided patches.

Remark 11. *The surface U_5 plays a key role in the theory of 5-sided patches. It would be interesting to investigate deeper geometric properties of U_5. Here are two of them: 1) as a surface in $\mathbb{R}P^5$ it contains 10 lines; 2) exactly 5 conics go through a generic point of U_5.*

A schematic of the interrelations between 5-sided patches is shown in Fig. 3.

§5. Toric Patches

Here we present several results about toric patches obtained in [8]. Some details can be found also in [16].

Consider a lattice $\mathbb{Z}^2$ of points with integer coordinates in the real affine plane $\mathbb{R}^2$. We call a convex polygon $\Delta \subset \mathbb{R}^2$ a lattice polygon if its vertices are in the lattice $\mathbb{Z}^2$. Edges δ_i of Δ define lines $h_i(\boldsymbol{t}) = \langle \boldsymbol{n}_i, \boldsymbol{t} \rangle + a_i = 0$, with inward oriented normal vectors $\boldsymbol{n}_i$, $i = 1, \ldots, r$. We choose $\boldsymbol{n}_i$ to be primitive lattice vectors, i.e. the shortest vectors with integer coordinates in the given direction.

Denote by $\widehat{\Delta} = \Delta \cap \mathbb{Z}^k$ a set of lattice points of the polygon Δ. It is easy to see that $h_i(\boldsymbol{m})$ is a non-negative integer for all $i = 1, \ldots, r$ and $\boldsymbol{m} \in \widehat{\Delta}$.

Definition 12. *A* toric patch *associated with a lattice polygon Δ is a rational patch T_Δ with a domain $D = \Delta$ and basis functions*

$$f_{\boldsymbol{m}} = c_{\boldsymbol{m}} h_1^{h_1(\boldsymbol{m})} h_2^{h_2(\boldsymbol{m})} \cdots h_r^{h_r(\boldsymbol{m})} \tag{6}$$

indexed by lattice points $\boldsymbol{m} \in \widehat{\Delta}$. Here $c_m > 0$ are some coefficients which may vary from case to case.

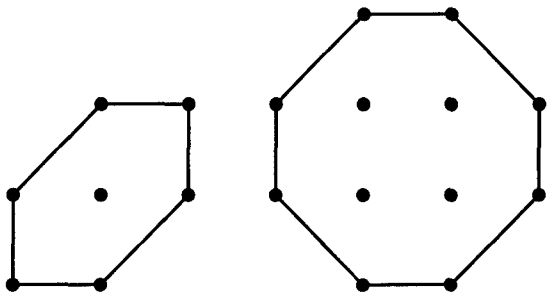


Fig. 4. Examples of lattice polygons.

Example 13. *Bézier surfaces and the Warren hexagon [13] are toric:*

1) *If Δ is a triangle with vertices $(0,0)$, $(d,0)$ and $(0,d)$, then T_Δ with $c_{(i,j)} = d!/(i!j!(d-i-j)!)$ is exactly a rational Bézier triangle of degree d, which parameter domain is scaled d times.*
2) *If Δ is a rectangle with four vertices $(0,0)$, $(d_1,0)$, (d_1,d_2) and $(0,d_2)$, then T_Δ with coefficients $c_{(i,j)} = \binom{d_1}{i}\binom{d_2}{j}$ is a tensor product surface of bidegree (d_1,d_2) with a scaled parameter domain $[0,d_1] \times [0,d_2]$.*
3) *Let Δ be a hexagon Δ_6 (see Fig. 4) then T_Δ with appropriate coefficients $c_{\boldsymbol{m}}$ is the Warren 6-sided patch denoted by $\widetilde{T}_6^1$ in Section 3.*

Toric patches have similar properties as Bézier surfaces. They are *affine invariant*, and have *convex hull property*. Every edge δ_i of the lattice polygon Δ corresponds to a boundary rational Bézier curve with control points $\boldsymbol{m} \in \widehat{\delta_i} = \delta_i \cap \mathbb{Z}$. In particular, its degree is equal to an 'integer length' of the edge δ_i.

The following property is in some sense similar to the affine invariance of the domain for Bézier surfaces.

Lemma 14. *(Unimodular invariance of the domain.) Let two lattice polygons be related via some affine unimodular transformation $L(\Delta) = \Delta'$ (i.e. L preserves the lattice $\mathbb{Z}^2$). Then toric patches T_Δ and $T_{\Delta'}$ with the same control points and weights are just reparametrizations of each other: $T_\Delta = T_{\Delta'} \circ L$.*

In Fig. 4 we see a lattice hexagon Δ_6 and an octagon Δ_8. Since they have 6- and 4-sided symmetry, corresponding toric patches T_Δ for $\Delta = \Delta_6, \Delta_8$ have the same symmetry.

Corollary 15. *For $m = 3, \ldots, 8$, the only symmetric (in the sense of Section 3) toric patches may be 3-, 4- and 6-sided, for example, Bézier triangles, tensor product surfaces of degree (d,d) and the Warren hexagon $T_{\Delta_6} = T_6^1$. In particular, the 5-sided T_5^n-patch cannot be toric.*

Proof: These numbers correspond to cyclic subgroups in the group $\mathrm{SL}_2(\mathbb{Z})$ of unimodular linear transformations of the lattice $\mathbb{Z}$. □

It is clear that an affine unimodular transformation L preserves area, since $\det L = \pm 1$. It is convenient to use so-called *normalized area* which is twice as large as the usual area in $\mathbb{R}^2$, since then area(Δ) is always integer for lattice polygons Δ. The following result is well-known in the theory of toric varieties (see [14] for an elementary proof).

Theorem 16. *The* implicit degree $\deg T_\Delta$ *of a toric patch* T_Δ *does not exceed* area(Δ). *It is equal to* area(Δ) *when the control points are in general position.*

For example, $\deg T_{\Delta_6} = 6$ and $\deg T_{\Delta_8} = 14$ (see Fig. 4). Consider now the more general parametrization of a toric patch $F' : \mathbb{R}^r_{\geq 0} \to \mathbb{R}^k$ defined as in (1) via basis functions

$$f'_{\boldsymbol{m}}(u_1, u_2, \ldots, u_r) = c_{\boldsymbol{m}} u_1^{h_1(\boldsymbol{m})} u_2^{h_2(\boldsymbol{m})} \cdots u_r^{h_r(\boldsymbol{m})}, \quad \boldsymbol{m} \in \Delta.$$

Definition 12 is obtained substituting variables u_i by affine forms h_i. Although the domain $\mathbb{R}^r_{\geq 0}$ has dimension r, the image of F' is 2-dimensional in all cases (cf. [8]). Hence, using various substitutions, one can get different interesting parametrizations of the same patch. The simplest piecewise substitution

$$\Phi_i(u, v) = F'(\underbrace{1, \ldots, 1}_{i-1}, u, v, 1, \ldots, 1), \quad i = 1, \ldots, r-1$$
$$\Phi_r(u, v) = F'(v, 1, \ldots, 1, u), \quad 0 \leq u, v \leq 1,$$

defines a subdivision of the toric patch into r tensor product pieces. This directly generalizes the Warren hexagon subdivision [13].

Acknowledgments. Both authors were partially supported by a grant from the Lithuanian Foundation of Studies and Science.

References

1. Gregory, J. A., N-sided surface patches, in *The Mathematics of Surfaces*, J. A. Gregory (ed.), Oxford University Press, 1986, 217–232.
2. Hosaka, M., and F. Kimura, Non-four-sided patch expressions with control points, Comput. Aided Geom. Design **1** (1984), 75–86.
3. Karčiauskas K., Rational m-sided surface patches, in *The Mathematics of Surfaces VIII*, R. Cripps (ed.), Information Geometers, 1998, 355–368.
4. Karčiauskas K., On five- and six-sided rational surface patches, preprint, 1999 (`http://www.mif.vu.lt/katedros/cs2/publicat/public.htm`).
5. Karčiauskas K., Rational m-sided Sabin–Hosaka–Kimura like surface patches, preprint, 1999 (`http://www.mif.vu.lt/katedros/cs2/publicat/public.htm`).
6. Karčiauskas K., Algebraic version of convex combination patches, in Proceedings of XL Conference of Lithuanian Mathematical Society, Technika, Vilnius 1999, 178–182.

7. Krasauskas R., New applications of real toric varieties in CAGD, Freiformkurven und Freiformflächen, Tagungsbericht 23/1998.
8. Krasauskas R., Toric surface patches I, preprint, 2000.
9. Loop, Ch., and T. DeRose, A multisided generalization of Bézier surfaces, ACM Trans. on Graphics **8** (1989), 204–234.
10. Loop, Ch., and T. DeRose, Generalized B-spline surfaces of arbitrary topology, Computer Graphics **24** (1990), 347–356.
11. Sabin M., Non rectangular surfaces for inclusion in B-spline surfaces, in *Eurographics'83*, T. Hagen (ed.), 1983, 57–69.
12. Sabin M., A symmetric domain for 6-sided patches, in *The Mathematics of Surfaces IV*, A. Bowyer (ed.), Clarendon Press, 1991, 185–193.
13. Warren J., Creating multisided rational Bézier surfaces using base points, ACM Trans. on Graphics **11** (1992), 127–139.
14. Warren, J. A bound on the implicit degree of polygonal Bézier surfaces, in *Algebraic Geometry and Applications*, C. Bajaj, (ed.), 1994, 511–525.
15. Zheng J. J., and A. A. Ball, Control point schemes over non-four-sided areas, Comput. Aided Geom. Design **14** (1997), 807–821.
16. Zubė, S., n-sided toric surface patches and A-resultants, Comput. Aided Geom. Design, submitted.

K. Karčiauskas and R. Krasauskas
Dept. of Mathematics and Computer Science
Vilnius University
Naugarduko 24, 2600 Vilnius, Lithuania
kestutis.karciauskas@maf.vu.lt
rimvydas.krasauskas@maf.vu.lt

N-sided Surface Generation from Arbitrary Boundary Edges

Kiyotaka Kato

Abstract. This paper discusses a general theory and an implementation method for generating a surface patch with concave edges, holes, ridges and valleys in CAD/CAM applications. The surface generation method, which has been proposed to create an N-sided patch with holes, is first reviewed. Such surfaces are generally classified as transfinite surfaces, in which a surface is interpolated to span given curves. In the proposed method, each boundary edge defined in a 2-dimensional domain has an appropriate blending function. The function is defined so that the derivatives are 0 on the edges, and the function values are 1 on one edge and 0 on the other edges, and each edge in the 3-dimensional space is blended smoothly. A revised method is also introduced in this paper. The previously proposed method has some problems in that a surface may not be generated appropriately for concave edges, and the surface has to be manipulated manually if it has holes. This causes distortion and overlap in mapping from a 2D domain into 3D space. In the new method, the blending function is revised, and the boundary edges in the 2D domain are obtained from the edges in the 3D space beforehand. Thus, it is shown that an N-sided patch with concave edges, holes, ridges and valleys can be suitably generated.

§1. Overview

It seemed that the study of surface generation was almost complete after the development of the NURBS (Non-Uniform Rational B-Spline) surface, and many commercial CAD system used the NURBS surface as a unified surface in their systems. However, it is now being recognized that the NURBS surface has some limitations, and is not suitable for some actual cases. In one such case, there is a problem with the generation of an n-sided surface patch. It is rather hard to generate a surface patch for arbitrary topology with the NURBS surface. Besides the development of 4-sided patches, n-sided surfaces have also been studied. The methods developed can be classified into three classes: the recursive subdivision method using polyhedrons, the multiple patch method in which a surface is represented by plural 4-sided patches, and the single surface

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 173–182.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

patch method in which a surface patch is represented as just one patch [1]. This paper refers to the single patch method.

A single patch method has been proposed which generates a surface patch from an arbitrary shape and a number of edges together with holes [2]. It was suggested that some relations are needed between the shape of the boundary edges and the shape of the 2D-definition boundary. The method of surface generation is a generalized one, but it was found that a surface cannot be generated well in some cases. An illegal surface is generated when the boundary has a concave shape and the surface has a hole. Distortion or overlap is caused in mapping from a 2D definition space into the 3D space.

Sabin calls the method of surface generation from boundary edges a "transfinite surface" in contrast with the one which is characterized by a finite number of control points. He argued a general theory, and proposed a two sided surface patch and a surface with holes [3-5]. The two sided surface interpolates two given Bezier curves in a 2D definition space so that it forms a smooth surface without singularity. He also tried to resolve this problem from 3D into 2D by using a dynamic model with some constraints to generate a surface patch with holes.

For this same purpose, this paper proposes a method of surface generation which is flexibile in generating a surface from such boundary edges so as not to cause twists and overlaps. The second section of this paper reviews the theory about pre-proposed surface generation. In the third section, the problems of the conventional method are discussed. After that, a new method of resolving these problems is described. After showing some examples of surface generation, the results are evaluated and conclusions are drawn.

§2. Surface Generation from Boundary Edges

2.1 General theory of surface generation

The fundamental idea is that a surface is created so that the interpolated point of a surface is obtained using rational blending functions for positional vectors and tangential vectors. Thus, the surface is a transfinite surface in consideration of the boundary positions and cross-boundary derivatives on the given boundary edges. It is a parametric surface created in mapping from a definition domain to 3D space $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. Let us call the definition domain Ω and the boundary of the domain Γ. Using points $\alpha \in \Omega, \beta \in \Gamma$, $\boldsymbol{B}(\beta)$ is a positional vector and $\boldsymbol{D}(\beta)$ is a cross boundary derivative of a given boundary edge. These vectors specify the boundary conditions and have to be given as follows. A surface is represented as in (2) using blending functions at a point P:

$$\boldsymbol{B}(\beta) = \lim_{\alpha \to \beta} \boldsymbol{S}(\alpha), \qquad \boldsymbol{D}(\beta) = \lim_{\alpha \to \beta} \partial \boldsymbol{S}(\alpha)/\partial n, \tag{1}$$

$$\boldsymbol{S}(\alpha) = \oint_{\Gamma} \Phi(\alpha)(\boldsymbol{B}(\beta) + |\alpha - \beta|\boldsymbol{D}(\beta))). \tag{2}$$

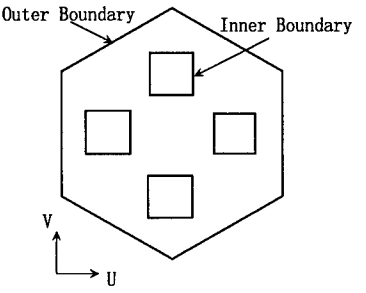


Fig. 1. Definition of the outer and inner boundaries.

2.2 Implementation method

Here the actual implementation method of N-sided surface-patch generation is described. Consider the normalized regular N-sided polygon in 2D space shown in Fig. 1 so that the foot length from an arbitrary point to each side is less than 1. This polygon is called "the outer boundary". Next, assume that one or more regular N-sided polygons are located within the outer boundary so that none of these polygons intersect with another. These polygons are called "inner boundaries". The closed domain D is defined as the area inside the outer boundary and outside the inner boundaries and is mapped to an N-sided surface patch in 3D space. Next prepare a pair of a boundary parameter and a distance parameter as follows.

(1) The distance parameter $d_{i,j}$ becomes 0 on side i,j, and varies from 0 to 1 according to the distance between point P and the side.

(2) The boundary parameter $b_{i,j}$ varies from 0 to 1 on side i,j of the given point P, and $b_{i,j}$ is given as the ratio of the adjacent distance parameters so that

$$b_{i,j} = d_{i,j-1}/(d_{i,j-1} + d_{i,j+1}). \tag{3}$$

Here i is the index of the outer boundary when $i = 0$, and the index of an inner boundary when $i \neq 0$. Here j is the index of a side of each boundary.

The values of u and v determine an arbitrary point $P(u,v)$ in the closed domain D. The pair of a distance parameter $d_{i,j}$ and a boundary parameter $b_{i,j}$ also determines the same point P. The blending function $\Phi_{l,m}$ for a side m of a boundary l is defined in the closed domain D as follows:

$$\Phi_{l,m}(u,v) = \frac{(1 - d_{l,m}^2)/d_{l,m}^2}{\sum_{p=0}^{H}\sum_{q=1}^{M_p}(1 - d_{p,q}^2)/d_{p,q}^2}. \tag{4}$$

Here the indices p and q relate to a side q of a boundary p. The boundary is the outer boundary when $p = 0$. Here H is the number of holes, and M_p is the number of sides of the boundary p. Also $\Phi_{l,m}$ is a function of u, v

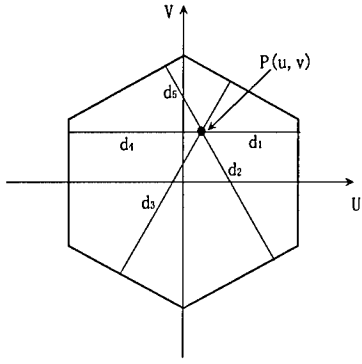


Fig. 2. Distance parameter for the outer boundary.

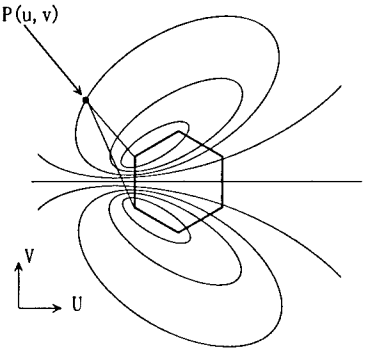


Fig. 3. Distance parameter and boundary parameter for an inner boundary.

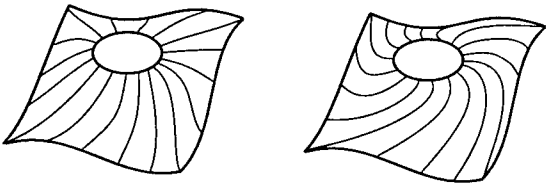


Fig. 4. Difference in surface generation according to definition domain.

and can also be a function of the pair of independent variables $b_{i,j}$ and $d_{i,j}$, because $d_{l,m}$ and $b_{l,m}$ are functions of $b_{i,j}$ and $d_{i,j}$. As shown in Fig. 2, the 3D boundary condition for a side j of a boundary i is given by a positional vector, $\boldsymbol{B}_{l,m}(b_{l,m})$ and a tangential vector, $\boldsymbol{D}_{l,m}(b_{l,m})$, where each vector is parameterized by the boundary parameter $b_{l,m}$. By giving the values of u and v, the variables $b_{p,q}$ and $d_{p,q}$ for side q of the boundary p are obtained, and $\boldsymbol{B}_{p,q}(b_{p,q})$ and $\boldsymbol{D}_{p,q}(b_{p,q})$ can be determined. The N-sided patch is given by the equation

$$\boldsymbol{S}(u,v) = \sum_{p=0}^{H} \sum_{q=1}^{M_p} \Phi_{p,q}(\boldsymbol{B}_{p,q}(b_{p,q}) + d_{p,q}\boldsymbol{D}_{p,q}(b_{p,q})). \tag{5}$$

§3. Problems with Conventional Surfaces

The previously proposed method removes the restrictions of a 4-sided patch [2]. This method enables a surface to be generated from the given boundary conditions (position, tangent vector), and is able to represent holes on the surface. However, it sometimes needs a manual transaction to generate a surface. For example, a generated surface sometimes becomes twisted or illegal when generating a surface with holes like the one shown in Fig. 4, although the

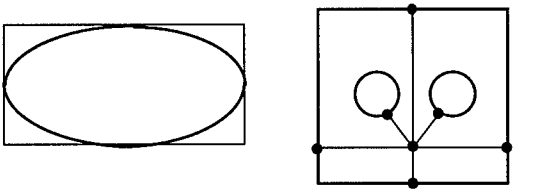


Fig. 5. Sabin's parametric domain.

surface satisfies the given boundary condition. Therefore, the earlier method must be revised in order to create a surface freely from arbitrary edges. It seems that the boundary in the 2D domain has to be similar to the 3D boundary in the 3D space. From such a point of view, Sabin proposed a surface patch using the definition space shown in Fig. 5. He also proposed a surface using dynamics from the same point of view with some restrictions from the same idea [5].

§4. Surface Generation from Boundary Edges by Reverse Mapping

In order to create a surface patch with concave edges and holes, the interpolation method has to be able to interpolate boundary edges on a plane at least without overlaps and protrusions. A transfinite surface blends sample points on the given boundary edges in an appropriate ratio. Therefore, the point to be interpolated exists in the convex hull of the sample points. Thus, it can be said that it is essentially difficult for a transfinite surface to generate a surface with a concave edge. Fig. 6a shows the boundary edges with a concave part on a plane in 3D space. It is desirable that point P is given as the point obtained by blending the sample points A1-A5 on each boundary edge. The point is in the convex hull of points A1-A5 because the blending functions have a value between 0 and 1.

Now consider the domain of Fig. 6b and Fig. 6c in UV-space. In case of Fig. 6b, the corresponding sample points b1-b5 are obtained for a point $p(u, v)$ and the sample points in 3D space become the points B1-B5 shown in Fig. 6d. It is thus possible that the blended point Q will be placed outside the boundary edges. In the case of Fig. 6c, sample points c1-c5 for point p will be obtained, and the corresponding points will be points C1-C5. Points C1 and C5 are affected strongly near the boundary, and point Q becomes an interpolated point and gives the good interpolated result shown in Fig. 6e. From the reasons outlined above, it can be said that the sample points are obtained near a point in the domain by using the definition boundary which resembles the given boundary edge. Since the sample points in 3D space for the point are given in appropriate ratios, a better interpolation can be realized compared with the conventional method in which a domain is a regular polygon. It can be also said that cross-boundary derivatives are helpful in avoiding a web in the concave part. The second term of (5) gives an effect of cross-tangential

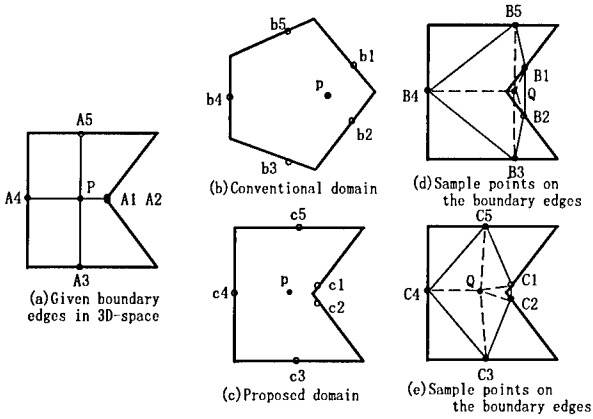


Fig. 6. Difference in sample points according to domain definition.

vectors to the surface. This term makes the surface point move according to the product of the distance parameter and the cross-boundary vector from the boundary.

4.1 Reverse mapping algorithm

In the previous section, it was stated that the boundary in a 2D definition space has to be similar to the shape of the 3D boundary edges. The following should be observed in constructing the domain boundary:

(1) The scale of 3D edges should reflect one of the domain edges.

(2) The topology of 3D edges should be the same as one of the domain edges.

(3) The angle of adjacent edges should reflect one of the domain edges.

Let us now consider a reverse mapping which satisfies these properties. Imagine a rubber surface spread over wires. By leaving it free, the wires will be straight. It would be ideal to use something like these wires as a 2D domain boundary. However this ideal mapping would be disadvantageous when considering the computational cost, so we selected a simple method of reverse mapping. Fig. 7 shows the algorithm. Each edge is connected at point P_k. Let the foot from the point to a plane be point Q_k. The plane can be obtained simply by solving the equation $\sum_{k=1}^{N} (P_k - P_k)^2$ so that it is minimized using the least square method. After obtaining the foot Q_k of the point to the plane, define the polygon which is constructed by Q_k as a 2D domain boundary.

4.2 Implementation method

In Section 2, the distance parameters and boundary parameters were defined in the normalized regular polygon of Fig. 2. The revised version of surface generation uses the same parameters, but uses non-normalized and irregular polygons with convex and concave parts. In order to define the distance

parameters for such polygons, we applied the previous calculation method used for the inner boundary to both the outer boundary and the inner boundaries. The boundary parameters are obtained by (3). Also the blending functions have to be revised because (4) is defined for the domain of a regular polygon. Next (7) is substituted for (5) in order to represent the concave figures shown in the next section:

$$\Phi_{l,m}(u,v) = \frac{1/d_{l,m}^2}{\sum_{p=0}^{H}\sum_{q=1}^{M_p} 1/d_{p,q}^2}, \tag{6}$$

$$\boldsymbol{S}(u,v) = \sum_{p=0}^{H}\sum_{q=1}^{M_p} \Phi_{p,q}(u,v)(\boldsymbol{B}_{p,q}(b_{p,q}) + d_{p,q}\boldsymbol{D}_{p,q}(b_{p,q},d_{p,q})). \tag{7}$$

4.3 Theorems

Some characteristics of the blending function and the surface patch defined in the previous section are now discussed. The following theorems can be obtained from (6) and (7). Select a pair of independent variables $b_{i,j}$ and $d_{i,j}$.

Theorem 1. $\lim_{d_{i,j}\to 0} \Phi_{i,j}(u,v) = 1$, $\lim_{d_{l,m}\to 0} \Phi_{i,j}(u,v) = 0$ *when* $(l,m) \neq (i,j)$.

Theorem 2. $\dfrac{\partial \Phi_{l,m}(u,v)}{\partial b_{i,j}} = 0$ *and* $\dfrac{\partial \Phi_{l,m}(u,v)}{\partial d_{i,j}} = 0$.

Theorem 3. $\lim_{d_{i,j}\to 0} \boldsymbol{S}(u,v) = \boldsymbol{B}_{i,j}(b_{i,j})$, *and*

$$\lim_{d_{i,j}\to 0} \frac{\partial \boldsymbol{S}(u,v)}{\partial b_{i,j}} = \frac{\partial \boldsymbol{B}_{i,j}(b_{i,j})}{\partial b_{i,j}}, \quad \lim_{d_{i,j}\to 0} \frac{\partial \boldsymbol{S}(u,v)}{\partial d_{i,j}} = \lim_{d_{i,j}\to 0} \boldsymbol{D}_{i,j}(b_{i,j},d_{i,j})$$

Proof:

$$\lim_{d_{i,j}\to 0} \frac{\partial \boldsymbol{S}(u,v)}{\partial b_{i,j}} = \lim_{d_{i,j}\to 0} \sum_{p=0}^{H}\sum_{q=1}^{M_p} [\frac{\partial \Phi_{p,q}}{\partial b_{i,j}}(\boldsymbol{B}_{p,q}(b_{p,q}) + d_{p,q}\boldsymbol{D}_{p,q}(b_{p,q},d_{p,q})) +$$
$$\Phi_{p,q}(\frac{\partial \boldsymbol{B}(b_{p,q})}{\partial b_{i,j}} + \frac{\partial d_{p,q}}{\partial b_{i,j}}\boldsymbol{D}_{p,q}(b_{p,q},d_{p,q}) + d_{p,q}\frac{\partial \boldsymbol{D}(b_{p,q},d_{p,q})}{\partial b_{i,j}}]$$
$$= \lim_{d_{i,j}\to 0} \Phi_{i,j}(\frac{\partial \boldsymbol{B}(b_{i,j})}{\partial b_{i,j}} + \frac{\partial d_{i,j}}{\partial b_{i,j}}\boldsymbol{D}_{i,j}(b_{i,j},d_{i,j}) + d_{i,j}\frac{\partial \boldsymbol{D}(b_{i,j},d_{i,j})}{\partial b_{i,j}}) = \frac{\partial \boldsymbol{B}(b_{i,j})}{\partial b_{i,j}}$$

$$\lim_{d_{i,j}\to 0} \frac{\partial \boldsymbol{S}(u,v)}{\partial d_{i,j}} = \lim_{d_{i,j}\to 0} \sum_{p=0}^{H}\sum_{q=1}^{M_p} [\frac{\partial \Phi_{p,q}}{\partial d_{i,j}}(\boldsymbol{B}_{p,q}(b_{p,q}) + d_{p,q}\boldsymbol{D}_{p,q}(b_{p,q},d_{p,q})) +$$
$$\Phi_{p,q}(\frac{\partial \boldsymbol{B}(b_{p,q})}{\partial d_{i,j}} + \frac{\partial d_{p,q}}{\partial d_{i,j}}\boldsymbol{D}_{p,q}(b_{p,q},d_{p,q}) + d_{p,q}\frac{\partial \boldsymbol{D}(b_{p,q},d_{p,q})}{\partial d_{i,j}}]$$
$$= \lim_{d_{i,j}\to 0} \Phi_{i,j}(\frac{\partial \boldsymbol{B}(b_{i,j})}{\partial d_{i,j}} + \boldsymbol{D}_{i,j}(b_{i,j},d_{i,j}) + d_{i,j}\frac{\partial \boldsymbol{D}(b_{i,j},d_{i,j})}{\partial d_{i,j}}) = \boldsymbol{D}_{i,j}(b_{i,j},d_{i,j}).$$

□

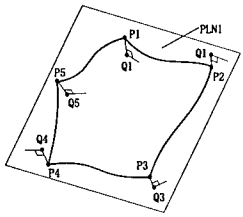


Fig. 7. Inverse mapping from 3D space into 2D domain.

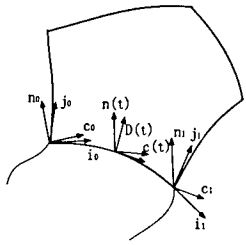


Fig. 8. Cross-boundary derivative.

§5. Boundary Condition and Surface Connection

5.1 Setting boundary conditions

A Coons patch must satisfy a compatibility condition. A Coons patch, which is constructed using a two surface patch in principle, does not guarantee the boundary condition without a compatibility condition. Gregory used a rational blending method, and invented a method of setting cross boundary derivatives freely so as to remove the inconvenience. A Gregory patch needs a compatibility condition for a positional boundary, and the twist vectors are discontinuous at the corners. In Little's patch the boundary conditions can be freely set for both position and tangent vector. The proposed surface does not cause the problems of the so-called compatibility condition. However, it is desirable that the tangent vector and the twist vector are continuous at the corners. Here, the method of setting cross-boundary derivatives at the corners is introduced. As shown in Fig. 8, two normal unit vectors $\boldsymbol{n}_0, \boldsymbol{n}_1$ are first calculated from the boundary derivatives $\boldsymbol{i}_0, \boldsymbol{j}_0, \boldsymbol{i}_1$, and $\boldsymbol{j}_1$. Here, the normal vector at a concave corner should be reversed. Define the two vectors $\boldsymbol{c}_0, \boldsymbol{c}_1$ obtained at the tips of the edge as $\boldsymbol{c}_0 = \boldsymbol{j}_0 \times \boldsymbol{n}_0, \boldsymbol{c}_1 = \boldsymbol{j}_1 \times \boldsymbol{n}_1$. The vectors $\boldsymbol{n}(t), \boldsymbol{c}(t)$ are from interpolating $\boldsymbol{n}_0, \boldsymbol{n}_1$ and $\boldsymbol{c}_0, \boldsymbol{c}_1$ respectively. Thus the cross boundary derivatives $\boldsymbol{D}(t)$ are given as $\boldsymbol{D}(t) = \boldsymbol{n}(t) \times \boldsymbol{c}(t)$.

In order to satisfy the compatibility condition for a tangent vector in a corner, the magnitude of a cross boundary vector has to been properly given. Since the boundary parameter $b_{i,j}$ for the edge(i,j) is defined as being between 0 and 1, the following reference has to be applied:

$$\left|\frac{\partial b_{i,j-1}}{\partial d_{i,j}}\right|_{\substack{d_{i,j}=0\\ b_{i,j}=0}} = \frac{1}{d_{i,j-2}}\Big|_{\substack{d_{i,j}=0\\ b_{i,j}=0}}, \qquad \left|\frac{\partial b_{i,j+1}}{\partial d_{i,j}}\right|_{\substack{d_{i,j}=0\\ b_{i,j}=1}} = \frac{1}{d_{i,j+2}}\Big|_{\substack{d_{i,j}=0\\ b_{i,j}=1}} \tag{8}$$

5.2 Boundary condition at concave corners and holes

A special transaction must be done in the cases of concave corners and holes so that the tangent vectors coincide with each other. As shown in Fig. 9, an edge is connected to the adjacent edges at concave and concave corners. Also an edge is connected to the adjacent edges at convex and concave corners in Fig. 10. For these cases, the cross boundary derivative has to be given as in

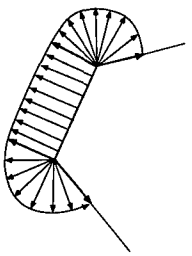


Fig. 9. Cross-boundary derivative in the concave-concave case.

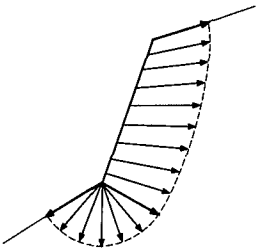


Fig. 10. Cross-boundary derivative in the convex-concave case.

these figures. Therefore, the derivative is given as a rational function. Giving the coefficients of (8) as $k0$ and $k1$ respectively, the tip vectors $\boldsymbol{c}_0, \boldsymbol{c}_1$ are given by (9). $\boldsymbol{c}_{00}$ and $\boldsymbol{c}_{11}$ are auxiliary vectors given to form the cross boundary vector. This is the reason why (5) is replaced with (7). For a surface patch with an isolated edge, the cross-boundary derivative can be set freely along the edge, because there is no adjacent surface around the isolated edge. Such a boundary edge is intended partially to trim a base patch:

$$\boldsymbol{c}_0 \equiv \frac{b\boldsymbol{c}_{00} + dk_0\boldsymbol{c}_0}{b + dk_0}, \qquad \boldsymbol{c}_1 \equiv \frac{(1-b)\boldsymbol{c}_{11} + dk_1\boldsymbol{c}_1}{(1-b) + dk_1} \tag{9}$$

§6. Example of Surface Generation

Figures 11 to 14 show examples of surface generation. Reverse mapping algorithms are applied to all of the surface generation. Compared with the old algorithm, it is unnecessary to modify the inner boundaries manually in a domain space. Fig. 11 shows an example with multiple holes. Fig. 12 shows an example with a ridge. The surface in Fig. 13 differs from the one in Fig. 14 in the shape of the hole, but both surfaces are generated in a desirable way.

§7. Conclusions

In this paper, a method of generating an N-sided patch with holes has been reviewed, and a revised method has been introduced. The following conclusions were obtained:

(1) The previous method has problems in generating a surface patch from boundary edges with concave parts and holes, because a transfinite surface essentially interpolates the sample points of given boundary edges.
(2) However, by using a reverse mapping from 3D space to 2D space, the shape of boundary edges becomes similar to the one in the 2D-space. This method relieves the above problems, and a surface can span arbitrary edges with holes and concave parts.
(3) In addition, this method can also represent isolated edges like ridges and valleys.

Acknowledgment. The author would like to thank Dr. M. A. Sabin of DAMPT, the University of Cambridge for his useful suggestions.

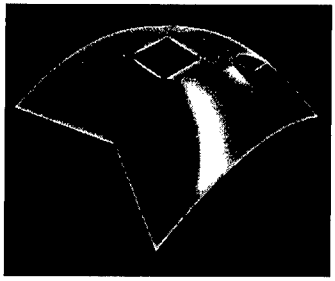


Fig. 11. Surface generation with two holes and a concave edge.

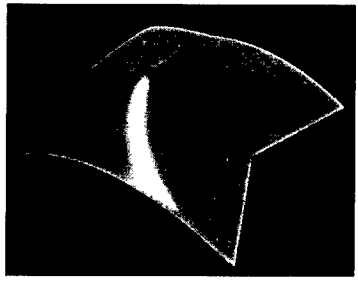


Fig. 12. Surface generation with a ridge.

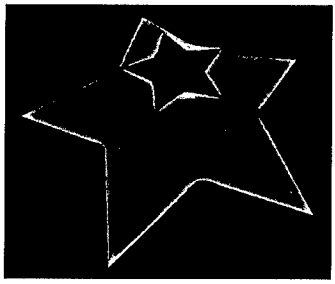


Fig. 13. Surface generation with concave parts and a hole.

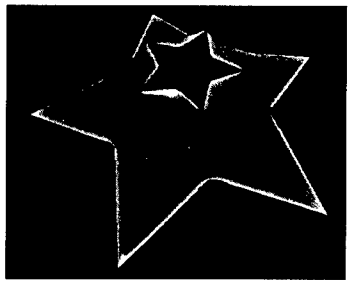


Fig. 14. The surface after the hole is rotated.

References

1. P. Malraison, A bibliography for N-sided surfaces, in Mathematics of Surfaces VIII Information Geometers, Winchester, 1998, 419–430.
2. K. Kato, Generation of N-sided surface patches with holes, Computer Aided Design **23** (1991), 676-683.
3. M. A. Sabin, Transfinite surface interpolation, in *The Mathematics of Surfaces VI*, G. Mullineux (ed.), Oxford, 1996, 517–537.
4. M. A. Sabin, Further transfinite surface developments, in Mathematics of Surfaces VIII, edited by Robert Cripps (ed.), Information Geometers, Winchester, 1998, 161-173.
5. M. A. Sabin, Two-sided patches suitable for inclusion in a B-Spline surface, in *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998, 409–416.

Mitsubishi Electric Corporation, Industrial Electronics and Systems Lab
8-1-1, Tsukaguchi-Honmachi, Amagasaki, Hyogo, Japan
kato@sdl.melco.co.jp

Advantages of Topological Tools in Localization Methods

Mohammed Khachan and Patrick Chenin

Abstract. Let $C = \{X \in \mathbb{R}^n \,/\, f(X) = 0\}$, $n \in \{2, 3\}$, where f is a polynomial function. We want to approximate C by subdividing the parameter space. Most of the usual algorithms raise two problems: data structure management, and the choice of subdivision level which respects the geometry of C. This paper gives a method based on a topological approach. In this work, we specify the local criteria that preserve the topological coherence between the model (the set C) and its volumetric approximation (the set of voxels that contains C). In addition, we determine the local criteria that give the digital analog of $(n-1)$ dimensional manifolds in $\mathbb{R}^n$. In this way, we determine locally how the set of voxels in digital space may be spread out to describe analogous properties of Euclidean manifolds. This gives efficient criteria for controlling the distribution of voxels and the depth of subdivision. We then obtain an approximation that conserves the topological properties of C. The process of localization based on these criteria is generated by an iterative mesh subdivision and skeleton.

§1. Introduction

In recent years, there has been a growing interest in using implicit surfaces for geometric modelling. Especially, the problem of constructing a polygonal approximation of implicit surfaces has received a great deal of attention. The basic idea of all methods for creating a polyhedral approximation of an object is an appropriate subdivision of the relevant space. Polygonalization algorithms typically query the implicit surface through spatial sampling. No preliminary information about the topology of the object is required, and only characterisation coordinates of points are used in the reconstruction. An early polygonalization algorithm for implicit surfaces is described in [2]. It samples the equation of the implicit surface over a three-dimensional rectangular grid of points and linearly interpolates polygons in regions where the function values change signs.

Hall-Warren in [4] and Dessarce-Chenin in [3] presented algorithms based on space subdivision in conjunction with a Bernstein-Bézier representation.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 183–192.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

The Bézier representation is used to exclude regions which cannot contain parts of a surface. Therefore, the algorithms are able to detect also small components which may be not detected by a simple sampling of the defining polynomial on grid points. Furthermore, due to the sampling, the class of algorithms may create new components and merge components.

We say that a polygonalization algorithm does not preserve topology if connected components are not preserved. Finally, we have to propose a nice definition for topological invariance.

In this paper we develop a topological approach related to digital topology theory. It is based on sampling in conjunction with the Bézier representation and a thinning process. The mesh evolution is controlled locally from topological criteria. As far as we know, no author has explored the use of digital topology to control the mesh subdivision for approximating an iso-surface.

The basic idea of our method can be expressed as follows: determination of local criteria for which the voxel set that localizes a given surface has the same geometry as this surface with topology preservation. The geometry is related to manifold properties. For a given subdivision level, if the set of voxels that localize the surface verifies these criteria, we say that the subdivision level reflects the geometric properties of the surface and it's over. Otherwise, we adopt an iterative process coupling two phases (subdivision phase and thinning phase) until the set of voxels represents a digital surface or the upper bound of the subdivision level is reached.

In Section 2, we develop our motivation to use digital topology in localization methods. Section 3 provides some useful definitions and notations related to $3D$-digital topology. In Section 4, we establish the link between Digital and Euclidean topology from the concept of continuous analogous. This enables us to translate properties of polyhedral manifold to digital space. We obtain an efficient and local criterion to determine if a set of voxels is a digital surface. In Section 5, we give criteria for topology preservation and describe the thinning process, provide a brief description of the global algorithm and present some experimental results in $2D$ and $3D$.

§2. Mesh Generation

An implicit surface is given by

$$f(x,y,z) = \sum_{i_k=0}^{n_k} a_{i_1,i_2,i_3} x^{i_1} y^{i_2} z^{i_3} = 0, \text{ where } k \in \{1,2,3\} \text{ and } a_{i_1,i_2,i_3} \in \mathbb{R}.$$

The surface consists of all real points (x,y,z) that verify the above equation. A geometric object is considered as a closed subset of $\mathbb{R}^3$ with the definition $f(x,y,z) \leq 0$, and is called a solid. The boundary of such object is a so-called implicit surface. There is a classification of points in $\mathbb{R}^3$ with respect to the solid. Let $p = (x,y,z)$ a point of $\mathbb{R}^3$. Then

- $f(p) < 0$, *if* p is inside the solid,
- $f(p) = 0$, *if* p is on the boundary of the solid,
- $f(p) > 0$, *if* p is outside the solid.

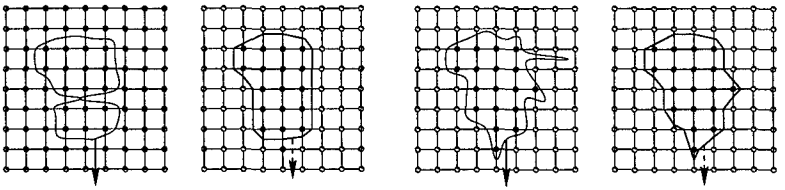


Fig. 1. The topology or the geometry of the initial surface is not preserved.

The classical approach consists of sampling the surface over the three dimensional rectangular grid, called voxels. The decision of whether a voxel is intersected by the surface is made by looking at the function values $f(p)$ at the eight vertices of the voxel. The surface intersects the voxel if not all signs of these values are equal. Thus, the combination of voxels intersecting the boundary of the solid provides an approximation of the whole original surface. Due to the sampling, this approximation cannot always have the same topology as the original surface (see Figure 1 (a,b)). The sign grid points criterion may miss information (see Figure 1 (c,d)): a voxel with all vertices of the same sign can intersect the original surface, but the criterion excludes this voxel class. Hence in order to preserve information, the criterion must keep all voxels that intersect the surface. The Bernstein-Bézier representation allows us to reach this objective.

First, we describe the implicit surface in the Bernstein-Bézier basis. Let $V = [a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$ be a voxel of the space subdivision. The polynomial $f(x,y,z) = \sum_{i_k=0}^{n_k} a_{i_1,i_2,i_3} x^{i_1}.y^{i_2}.z^{i_3}$ with $(x,y,z) \in V$, can be written in Bernstein-Bézier basis form as

$$f(x,y,z) = \sum_{i_k=0}^{n_k} b_{i_1,i_2,i_3}\ B_{i_1}^{n_1}(u).B_{i_2}^{n_2}(v).B_{i_3}^{n_3}(w),$$

with $(u,v,w) \in [0,1]^3$, and $B_l^n(x) = C_l^n.(1-x)^{n-l}.x^l$. The coefficients $b_{i,j,k}$ in $\mathbb{R}$ are the Bernstein-Bézier ordinates. There is a unique set of Bernstein-Bézier ordinates associated with each voxel V; we denote it by $P.C(V)$.

Initially, we have one voxel containing the iso-surface. When we subdivide the voxel in the three directions (x,y,z), the associated Bernstein-Bézier ordinates set is also subdivided (de-Casteljau subdivision) according to the three directions and we relate each new control-polygon with its associate voxel.

Let V be a voxel of the space subdivision. If all the elements of $P.C(V)$ have the same sign, then by the convex-hull property of the Bernstein-Bézier polynomial, f has the same sign over the entire voxel V. Three types of voxels can be found:

1) Outside voxels with $\forall p \in P.C(V),\ p > 0$.
2) Boundary voxels with $\exists p, q \in P.C(V),\ p.q \leq 0$.
3) Inside voxels with $\forall p \in P.C(V),\ p < 0$.

From this partition, we can generate two classes of voxels with respect to the solid, as follows:

- 1-voxel, which corresponds to a boundary voxel,
- 0-voxel, which corresponds to outside or inside voxel.

The convex-hull property of the Bernstein-Bézier polynomial implies that a 0-voxel cannot intersect the surface, but does not assert that a 1-voxel intersects the surface. So, the approximation cannot always have the same topology as the original surface (we can create components that do not exist in the original surface and merge components).

In order to overcome this problem, we develop a topological approach related to digital topology theory. It consists of representing the space subdivision by a binary three dimensional digital image. The 0-voxel represents a voxel of the background, and the 1-voxel a voxel of the image object. In order to avoid having to consider the boundary of the 3-digital image, we assume that the digital image is unbounded in all directions.

Let λ be a centroid-map which associates to each voxel σ its barycentre $\lambda(\sigma)$. λ is an one-to-one map between the set of all voxels in $\mathbb{R}^3$ and its associate digital grid $\mathbb{Z}^3$. Points of $\mathbb{Z}^3$ associated with 1-voxels are called black points, and those associated with 0-voxel are called white points. The set of black points normally corresponds to an object in the digital grid.

In the next section we recall the definition of a binary three-dimensional digital grid.

§3. Basic Notions in 3D-digital Grids

A significant concept in the study of a digital grid is that of neighborhood. By means of the neighborhood we are able to define "topology" in the digital space. A point $p \in \mathbb{Z}^3$ is defined by $(x_i(p))_{i=1}^3$ with $x_i(p) \in \mathbb{Z}$. We consider two types of neighbors of p in the $3D$-Digital Grids

- The 6-neighbors: $\mathcal{N}_6(p) = \{q \in \mathbb{Z}^3 \ : \ \sum_{i=1}^{3} |x_i(p) - x_i(q)| = 1\}$,
- The 26-neighbors: $\mathcal{N}_{26}(p) = \{q \in \mathbb{Z}^3 \ : \ \max_{1 \le i \le 3} \left(|x_i(p) - x_i(q)|\right) = 1\}$.

Let $\beta \in \{6, 26\}$ and $T \subset \mathbb{Z}^3$. We say that q is β-adjacent to p *if and only if* $q \in \mathcal{N}_\beta(p)$. p is said to be β-adjacent to T if p is β-adjacent to some point in T. Two sets T and W are said to be β-adjacent to each other if some point in T is β-adjacent to some point in W.

A β-path of length m, $m > 0$, from p to q in T means a sequence of distinct points $p = p_0, \ldots, p_m = q$ of T such that p_i is β-adjacent to p_{i+1}, $1 \le i < m$. Two points $p, q \in T$ are β-connected in T if and only if there exists a β-path from p to q in T. The equivalence classes of T under β-connectivity are called β-components of T. A set of points T is called β-connected if and only if every two points p, q in T are β-connected in T.

- $\mathcal{N}_T(p)$ denotes the set $\mathcal{N}(p) \cap T$.
- $\mathcal{N}_{\beta,T}(p)$ denotes the set of elements in T that are β-adjacents to p,

- T^c denotes the complement of T in $\mathbb{Z}^3$; $T^c = \mathbb{Z}^3 - T$.

We let $\mathcal{N}(p)$ denote the 27 points in the $(3,3,3)$ neighborhood of p. A point p in T such that $\mathcal{N}(p) \subset T$ is called an `interior` point, otherwise p is called a border point [10].

In this paper, our neighborhood structure corresponds to the $(6,26)$-adjacency relation: 6-adjacency for the object and (26)-adjacency for its complement in the digital image.

§4. Continuous Analogs and 2-digital Manifold in $\mathbb{Z}^3$

The notion of continuous analogs was introduced in [9] to establish general properties of binary three dimensional images and used in [8] to give a natural proof of a theorem on simple surface points.

Generally, this tool permits us to relate digital topology to polyhedral topology. In [7] we generalize the concept of continuous analogs in all dimension n, in the context of $(2n, 3^n - 1)$-adjacency, and establish the link between digital and Euclidean topology.

In the following, we recall some results in three dimension space given in [7] for all dimension space. Let T be a subset of $\mathbb{Z}^3$. We will construct a polyhedral complex $C(T)$ of T as follows:

- A 0-`cell` of $C(T)$ is an element of T,
- A 1-`cell` of $C(T)$ is an unit segment whose vertices belong to T,
- A 2-`cell` of $C(T)$ is an unit square whose vertices belong to T,
- A 3-`cell` of $C(T)$ is an unit cube whose vertices belong to T.

$C(T)$ is a complex, it is called the `cubical-complex` of T. The underlying space $|C(T)|$ is called the `continuous analog` of T in $\mathbb{R}^3$. Note that [5] gives a general method for generating polyhedra from a set of lattice points in $\mathbb{Z}^3$. The following theorem expresses the fundamental properties of continuous analogs. The proof is given in [7].

Theorem 1. *Let T be a subset of $\mathbb{Z}^3$.*

1) $|C(T)| \cap \mathbb{Z}^3 = T$,
2) *Two elements of T are in the same 6-component of T if and only if they are in the same component of $|C(T)|$,*
3) *Two elements of T^c are in the same 26-component of T^c if and only if they are in the same component of $\mathbb{R}^3 - |C(T)|$.*

The remainder of this section deals with the relation between a $2D$-digital surface, a $2D$-polyhedral manifold and the Jordan-Brouwer Theorem. In $3D$ Euclidean space, a simple closed surface is well defined: the neighborhood of each point in the surface is homeomorphic to an Euclidean disc. The analog property in digital space consists of caracterizing a 'surface' in $\mathbb{Z}^3$ by considering its associate continuous analog set.

Morgenthaler and Rosenfeld in [12] have introduced the notion of simple closed digital surface in order to establish a nontrivial $3D$-analog of the $2D$-Jordan curve theorem. They characterized a simple closed digital surface as a

connected collection of *orientable* simple surface points. In [8] T.Y. Kong and A.W. Roscoe reveal what simple surface points 'look like'. In [6], we establish in all dimensions the relation between n-dimensional digital manifold and n-dimensional Euclidean manifold. Here, we give only properties related to the caracterisation of a digital surface. General results and proof are given in [6].

Definition 2. *Let T be a subset of $\mathbb{Z}^3$ and $p \in T$. p is called a* simple surface point *of T if*

- *$\mathcal{N}_{T^c}(p)$ admits exactly two 26-components, denoted by $Int(p)$ and $Ext(p)$,*
- *$\forall q \in \mathcal{N}_{6,T}(p)$, q is 6-adjacent to $Int(p)$ and $Ext(p)$.*

Definition 3. *Let T be a subset of $\mathbb{Z}^3$ and $p \in T$. T is called a* digital surface *if $p \in T$, p is a simple surface point of T.*

Theorem 4. *Let T be a subset of $\mathbb{Z}^3$.*

- *T is a digital surface if and only if $|C(T)|$ is a simple and closed surface in $\mathbb{R}^3$.*
- *if T is a digital surface, then T^c has exactly two 26-components ($Int(T)$ and $Ext(T)$), and every element of T is 26-adjacent to these components.*

Let T be the set of black points in the digital grid. The notion of simple surface point gives an efficient and local criterion to determine if T is a digital surface. This criterion requires only a small number of local operation per point of T. By translating this characterisation in the voxel space, we obtain an efficient criterion to determine if a set of voxels represent a digital surface.

§5. Algorithm and Results

For a given subdivision level, the above criterion allows us to test if the set of 1-voxels represents a digital surface. If it is true, we say that the current subdivision level reflects the geometric properties of the original surface and it's over, otherwise our approach consists of two phases.

During the first phase, we subdivide the set of 1-voxels in the three directions and label the new voxels (1-voxel and 0-voxel). During the second phase, we use a thinning process (remove 0-voxels with topology preservation). We adopt a sequential thinning process: the border voxels with 0 values are 'peeled off' layer-by-layer with topology preservation. The remaining digital set, called the skeleton, contains all 1-voxels of the current level and will contain 0-voxels whose deletion would destroy the current topology.

The second phase permits us to control the topology evolution. We can merge components of the skeleton's complementary set or create a hole in the digital image, by removing a specific 0-voxel (if there exists) from the skeleton. The end step of this phase is to label all the voxels of the skelton to 1 (1-voxel).

Topological thinning is a widely used approach for generating skeletons from binary objects. It has been shown (see [10,1]) that the topology in a digital grid will be preserved by a thinning process if the border points that are removed during each step are simple points. The notion of simple point is related to topology preservation. A border point is a simple point *if and*

only if the Euler number and the number of connected components in its neighborhood does not change after its removal [10,6,11].

The body of our method, called the treatment phase, is organised in an iterative way, each step consists of the two phases described above, until the skeleton represents a digital surface or the upper bound of the subdivision level is reached.

Input

- Equation of the Implicit Surface, $f(x, y, z)$,
- Cuboid containing the surface.

Ouput

- A set of voxels that localizes the surface with topology and geometry preservation.

We begin by tranforming f in the Bernstein-Bézier basis related to the Cuboid. Our algorithm consists of two phases: initialization phase and treatment phase.

> During the initialization phase, we extract the components of the object and its complement (Interior and Exterior components).
>
> During the treatment phase, we apply iteratively the thinning and subdivision process until the skeleton represents a digital surface or the upper bound of the subdivision level is reached.

The subdivision process consists of combining the voxel space subdivision with control polygon subdivision. The thinning phase proceeds in an iterative way

- Update the border component of the object,
- For each border component, remove sequentially 0-voxels that correspond to a simple point.

We note that this work remains valid for any notion of digital surface satisfying the Jordan-Brouwer theorem.

5.1 Illustration in 2D case

We consider the following function written in terms of the Bernstein-Bézier basis:

$$\begin{aligned} f(x,y) &= 4700 - 40670x - 5000y + 160965x^2 + 5000y^2 - 264750x^3 + 155584x^4 \\ &= \sum_{i=0}^{2}\sum_{j=0}^{4} p_{ij} B_i^4(x) B_j^4(y). \end{aligned}$$

This example illustrates the concept of our approach. If the surface has no-singularity, our method gives the subdivision level for which the set of 1-voxels corresponds to a digital surface. The following example illustrates the

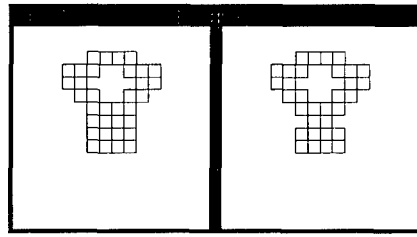


Fig. 2. Initialization phase.

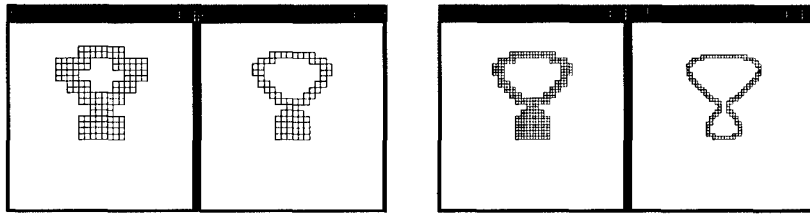


Fig. 3. Treatment phase.

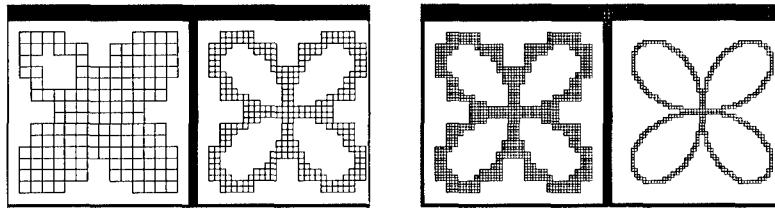


Fig. 4. Initialization and first step of treatment phase.

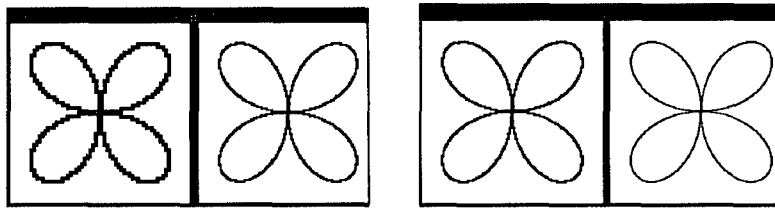


Fig. 5. Final result.

case where the initial object admits a singularity. Let

$$\begin{aligned} f(x,y) = {} & 4 - 32x + 128x^2 - 256x^3 + 288x^4 - 192x^5 + 64x^6 \\ & + 128xy - 320x^2y - 320xy^2 + 384x^3y + 512x^2y^2 + 384xy^3 \\ & - 192x^4y - 384x^3y^2 - 384x^2y^3 - 192xy^5 - 32y \\ & + 128y^2 - 256y^3 + 288y^4 - 192y^5 + 64y^6. \end{aligned}$$

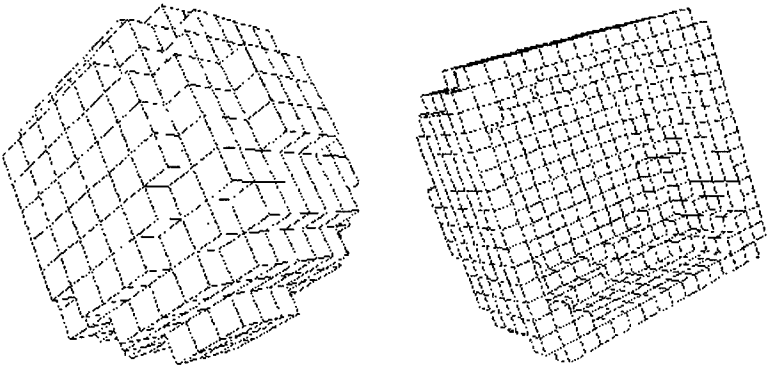


Fig. 6. A digital sphere.

5.2 Illustration in 3D case

Figure 6 illustrates the algorithm's application to a sphere given by an implicit equation. It shows external and internal sights.

§6. Conclusion

In this paper we use a digital topology approach to preserve the topological coherence between the model (original surface) and its volumetric approximation. Our method is based on subdivision and a thinning process. We use local and efficient criteria to determine the nature of the approximation and to preserve the current topology.

Acknowledgments. The authors would like to thank Hafsa Deddi and Rémi Dessarce for their help in the writing of this paper and for their numerous remarks and suggestions.

References

1. Bertrand, G., Simple points, topological numbers and geodesic neighborhoods in cubic grids, Pattern Rec. Letters **15** (1994), 1003–1011.
2. Bradshaw, C. B., Surfaces of functions of three variables, Master's thesis, Department of Civil Engineering, Brigham Young University, 1982.
3. Dessarce, R., Calculs par lancer de rayons, Thesis, Université Joseph Fourier, Grenoble-France, 1996.
4. Hal and Warren, Adaptative polygonalization of implicitly defined surfaces, IEEE Computer Graphics and Applications **10** (1990), 33–42.

5. Kenmochi, Y., A. Imiya, and N. Ezquerra, Polyhedra generation from lattice points, Sixth International workshop, D.G.C.I., Lectures Notes in C.S. 1176, Springer, 1996, 127–138.
6. Khachan, M., Etude topologique pour la localisation et la reconstruction d'objets géométriques, Thesis, Université Joseph Fourier, Grenoble, France, 1998.
7. Khachan, M., P. Chenin, and H. Deddi, Continuous analog concept and adjacency graph in n-digital image, Comput. Vision Graphics Image Process, submitted.
8. Kong, T. Y., and A.W. Roscoe, Continuous analog of axiomatized digital surfaces, Computer Vision, Graphics, and Image Processing **29** (1985), 60–86.
9. Kong, T. Y., and A. W. Roscoe, A theory of binary digital pictures, Computer Vision, and Image Processing **32** (1985), 221–243.
10. Kong, T. Y., and A. Rosenfeld, Survey digital topology: Introduction and survey, Computer Vision, Graphics, and Image Processing **48** (1989), 357–393.
11. Lee, C., T., Poston, and A. Rosenfeld, Winding and Euler numbers for 2D and 3D digital images CVGIP **53** (1991), 522–537.
12. Morgenthaler, D. G., and A. Rosenfeld, Surfaces in three-dimensional digital images, Information and Control **26** (1974), 24–33.
13. Rosenfeld, A. Three-dimensional digital topology, Information and Control **50** (1981), 119–127.

Mohammed Khachan
L.E.R.I (Laboratoire d'Etudes et de Recherches Informatique)
IUT-Leonard de Vinci, 51.059 Reims, France
khachan@leri.univ-reims.fr

Patrick Chenin
LMC-IMAG. Université Joseph Fourier
BP 53, 38.041 Grenoble, cedex 9, France
pchenin@imag.fr

Fast Voronoi Diagrams and Offsets on Triangulated Surfaces

Ron Kimmel and James A. Sethian

Abstract. We apply the Fast Marching Method on triangulated domains to efficiently compute Voronoi diagrams and offset curves on triangulated manifolds. The computational complexity of the proposed algorithm is optimal, $O(M \log M)$, where M is the number of vertices. The algorithm also applies to weighted domains in which a different cost is assigned to each surface point.

§1. Introduction

Voronoi diagrams play important roles in many research fields such as robotic navigation and control, image processing, computer graphics, computational geometry, pattern recognition, and computer vision. Its Euclidean version, for which there is an efficient implementation, is a building block in many applications.

The Voronoi diagram sets boundaries between a given set of source points, and splits the domain into regions such that each region corresponds to the closest neighborhood of a source point from the given set. Let our domain be D, let the set of given n points be $\{p_j \in D, j \in 0,..,n-1\}$, and the distance between two points $p, q \in D$ be $d(p,q)$. Then the Voronoi region G_i corresponds to the set of points $p \in D$ such that $d(p,p_i) < d(p,p_j), \forall j \neq i$.

Offsets computation is often used in approximation and singularity theories, and comes into practice in computer aided design (CAD) and numerical control (NC machines). Given a curve and its embedding space, an offset curve is defined by a set of points with a given fixed distance from the original curve.

There are some numerical and topological difficulties, even in the computation of offsets for curves in the 2D Euclidean plane, e.g. the formation of singularities in the curvature, self intersection of the offsetting curve, and the fact that an offset of a polynomial parametrized curve is not necessarily polynomial. Some of the numerical difficulties were addressed in [9], where

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 193–202.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

the Osher-Sethian level set method [16,20], which grew out of Sethian's earlier work on curve evolution, see [21], was used to overcome the topological changes.

Efficient construction of distance maps, minimal geodesics, Voronoi diagrams, and offset curves for non-flat and weighted domains is a challenging problem, see e.g. [15,13,8,12,6,10]. The core of our approach is Sethian's Fast Marching Method, [22,19,20] which solves the Eikonal equation on a rectangular orthogonal mesh in $O(M \log M)$ steps, where M is the total number of grid points. Contingent upon the triangulated upwind and monotonic update schemes given by Barth and Sethian [1], this technique was extended to triangulated surfaces by Kimmel and Sethian in [11]. The triangulated version of the Fast Marching Method has the same computational complexity, and solves the Eikonal equation on triangulated domains in $O(M \log M)$ steps, where M is the number of vertices. Using this technique, one can compute distances on curved manifolds with local weights. For other applications which rely on the Fast Marching Method, see [14,4].

Here we apply our method to compute Voronoi diagrams of a given set of points (or regions), and to find offsets from curves and points on triangulated manifolds. The computational complexity of the proposed algorithm is optimal $O(M \log M)$, its implementation is simple, and it also applies to weighted domains in which a different cost is assigned to each surface point.

The key idea is based on upwind finite difference operators as numerically consistent approximation to the differential operators in the Eikonal equation. Such an approximation selects the correct viscosity solution. The upwind operators allow us to construct a solution to the Eikonal equation by optimally sorting the updated points using a heap structure.

The outline of this paper is as follows. The key for fast computation of offsets and Voronoi diagrams is a fast algorithm for computing the distance. Hence, we first comment on the connection between the Eikonal equation and distance maps on weighted domains. We refer the reader to Sethian's Fast Marching Method for solving the Eikonal equation and for computing distance maps on orthogonal grids, and to [11] for details on our extension for computing the solution on triangulated domains. We then apply the method for the computation of fast Voronoi diagrams and offsets on triangulated manifolds.

§2. Fast Marching Method and the Eikonal Equation

We first explore some aspects of distance computation on weighted domains. In order to compute the distance between two points, we need to define a measure of length. A definition of an arclength allows us to measure distance by integrating the arclength along a curve connecting two points. The distance between the points corresponds to the length of the shortest curve connecting them.

Given a 2D weighted flat domain, or in other words an isotropic nonhomogeneous domain, the distance may be defined via the arclength definition. For

example, the arclength may be written as a function of the x and y Cartesian coordinates of the planar domain

$$ds^2 = \mathcal{F}(x,y)^2(dx^2 + dy^2),$$

where $\mathcal{F}(x,y) : R^2 \to R^+$ is a function that defines a weight for each point in the domain.

The distance map $T(x,y)$ from a given point p_0 assigns a scalar value to each point in the domain that corresponds to its distance from p_0. It is easy to show, see e.g. [2], that the gradient magnitude of the distance map is proportional to the weight function at each point

$$|\nabla T(x,y)| = \mathcal{F}(x,y),$$

where $|\nabla T| \equiv \sqrt{T_x^2 + T_y^2}$. This equation is known as the Eikonal equation. The 'viscosity' solution to the Eikonal equation coupled with the boundary condition $T(p_0) = 0$ results in the desired distance map.

Our first goal is to solve the Eikonal equation. The key is to construct a numerical approximation to the gradient magnitude that selects an appropriate 'weak solution'. Consider the following upwind approximation to the gradient, given by

$$\left(\max(D_{ij}^{-x}T, -D_{ij}^{+x}T, 0)^2 + \max(D_{ij}^{-y}T, -D_{ij}^{+y}T, 0)^2\right)^{1/2} = \mathcal{F}_{ij},$$

where for example $D_{ij}^{-x}T \equiv \frac{T_{ij} - T_{i-1,j}}{h}$ is the standard backwards derivative approximation, and $T_{ij} \equiv T(i\Delta x, j\Delta y)$. The use of upwind schemes in hyperbolic equations is well known, see for example, Godunov's paper from 1959 [7]. For Hamilton-Jacobi equations, see e.g. [17,3].

The solution T can be systematically constructed in an upwind fashion. The upwind difference approximation of the above equation means that information propagates one way from smaller values of T to larger values. The Fast Marching Method exploits this order of events. A point gets updated only by points with smaller values. This 'monotone property' allows us to keep a front of candidate points that tracks the flow of information, ordered in a heap tree structure in which the root is always the smallest value. An update of an element in the heap tree is done in $O(\log M)$ operations. Thereby, the total computational complexity is $O(M \log M)$. We refer to [22,19,20] for further details on the Fast Marching Method.

One could recognize similarity to Dijkstra's method [5,18] that computes minimum costs of paths on networks. Dijkstra algorithm would obviously fail to consistently solve our geometric problems. Actually, any graph-search-based algorithm induces the artificial metric imposed by the graph network, and would be inconsistent with the continuous case, and thus fail to converge as the graph resolution is refined.

The Fast Marching Method that works for orthogonal grids may be viewed as a selection for the update of one of the four right angle triangles that share

the same vertex. The extension to triangulated domains is motivated by this observation, by the geometric interpretation of the update step, and by an additional special treatment of obtuse angles. We refer to [11] for details on the extension of the fast marching method to triangulated domains. It is also based on a finite difference approximation to the Eikonal equation, this time on the surface, monotone by construction, consistent, upwind, and converges to the viscosity solution.

§3. Offsets and Voronoi Diagrams

We have an algorithm to compute distances on triangulated manifolds, and hence construct offset curves. First, we solve the Eikonal equation with speed $\mathcal{F} = 1$ on the triangulated surface to compute the distance from a source point or a region that defines an initial curve. We then find the equal geodesic distance curves on the surface by interpolating the intersection with a constant threshold using a 'marching triangle' procedure, again an $O(M)$ operation. The offsets on the triangulated surface, or the equi-geodesic-distance curves, are shown in Figure 1. The black curve is the original curve, and the white curves are the offsets.

Figure 2 presents Voronoi diagrams on several beads and a synthetic head. We first compute the distance from each of the initial given source points simultaneously using a single heap structure, and allow one vertex overlap between distance maps form different sources. The complexity for the distance computation is still $O(M \log M)$. Next, we 'march' along the triangles, and for each triangle linearly interpolate the intersection curve between the two different distance maps, again an $O(M)$ operation.

The algorithm complexity remains the same as we add weights to the surface. In Figures 3 and 4 a different cost is assigned to each vertex. The cost, or weight function, is texture mapped onto the triangulated surface. The weighted offsets, or weighted equal geodesic distance contours are shown in Figure 3, while weighted geodesic Voronoi diagrams for several surfaces are presented in Figure 4. In both examples, dark intensity mapped onto the surface indicates a low cost, and the brighter the intensity the higher the cost.

Acknowledgments. Calculations were performed at the University of California at Berkeley, the Lawrence Berkeley Laboratory, and the Technion, Israel Institute of Technology. This work was supported in part by the Applied Mathematics Subprogram of the Office of Energy Research under contract DE–AC03–76SF00098, and the National Science Foundation and Office of Naval Research under contract FDN00014-96-1-0381. RK is a Samuel & Miriam Wein Academic Lecturer, his research was partially supported by the fund for the promotion of research at the Technion.

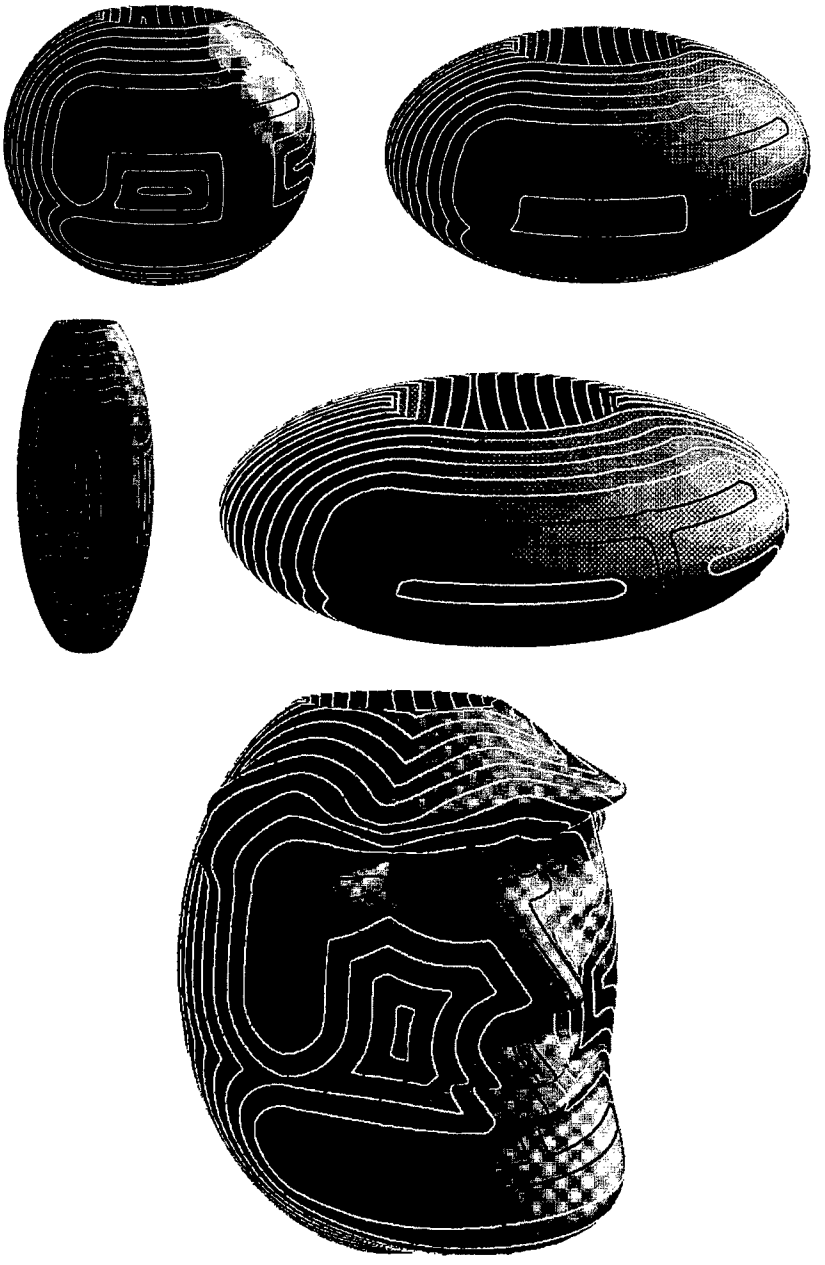


Fig. 1. Offsets on four beads and a Synthetic Head.

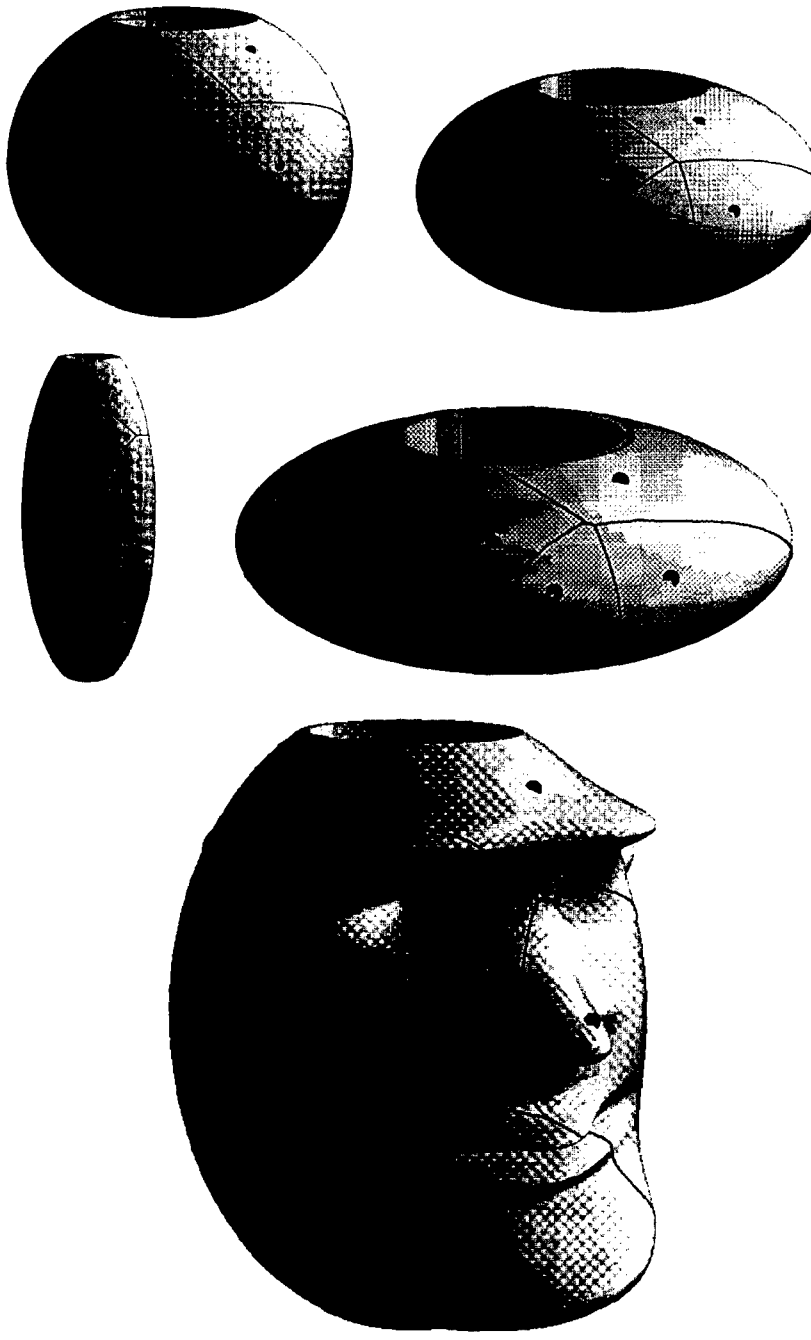


Fig. 2. Voronoi diagrams of five points on four beads and a Synthetic Head.



Fig. 3. Weighted offsets on four beads and a Synthetic Head.

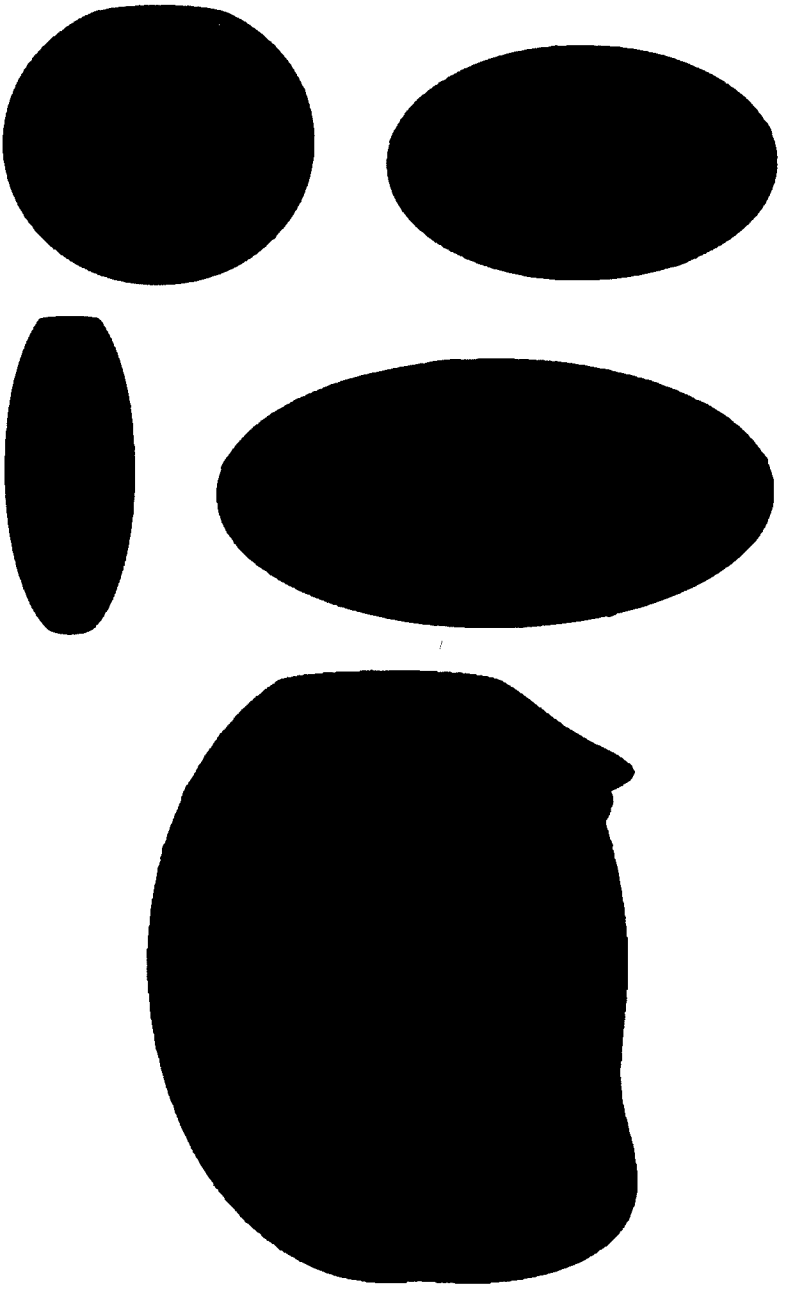


Fig. 4. Weighted Voronoi diagrams of five points on four beads and a Synthetic Head.

References

1. Barth, T., and J. A. Sethian, Numerical schemes for the Hamilton-Jacobi and level set equations on triangulated domains, J. Comp. Phys. **145**(1) (1998), 1–40.
2. Bellman, R., and R. Kalaba, *Dynamic Programming and Modern Control Theory*, London mathematical society monographs, London, 1965.
3. Brockett, R. W., and P. Maragos, Evolution equations for continuous–scale morphology, in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, 1992, 1–4.
4. Cohen, L. D., and R. Kimmel, Global minimum for active contours models: A minimal path approach, International Journal of Computer Vision **24**(1) (1997), 57–78.
5. Dijkstra, E. W., A note on two problems in connection with graphs, Numerische Mathematic **1** (1959), 269–271.
6. Elber, G., and E. Cohen, Error bounded variable distance offset operator for free form curves and surfaces, Int. J. Comput. Geom. & Applic. **1**(1) (1991), 67–78.
7. Godunov, S. K., Finite difference method for numerical computation of discontinuous solution of the equations of fluid dynamics, Matematicheskii Sbornik **47** (1959), 271.
8. Kimmel, R., A. Amir, and A. M. Bruckstein, Finding shortest paths on surfaces using level sets propagation, IEEE Trans. Pattern Anal. and Machine Intelligence **17**(6) (1995), 635–640.
9. Kimmel, R., and A. M. Bruckstein, Shape offsets via level sets, Computer-Aided Design **25**(5) (1993), 154–162.
10. Kimmel, R., N. Kiryati, and A. M. Bruckstein, Sub-pixel distance maps and weighted distance transforms, Journal of Mathematical Imaging and Vision, Special Issue on Topology and Geometry in Computer Vision **6** (1996), 223–233.
11. Kimmel, R., and J. A. Sethian, Computing geodesic paths on manifolds, Proceedings of National Academy of Sciences, USA, **95**(15) (1998), 8431-8435.
12. Kunze, R., F. E. Wolter, and T. Rausch, Geodesic Voronoi diagrams on parametric surfaces, Welfen lab. report, Welfen Lab., Univ. Hannover, Germany, July 1997.
13. Maekawa, T., Computation of shortest paths on free-form parametric surfaces, Journal of Mechanical Design **118** (1996), 499–508.
14. Malladi, R., and J. A. Sethian, An O(N log N) algorithm for shape modeling, Proceedings of National Academy of Sciences, USA, **93** (1996), 9389–9392.

15. Mitchell, J. S. B., D. M. Mount, and C. H. Papadimitriou, The discrete geodesic problem, SIAM J. Comput. **16**(4) (1987), 647–668.
16. Osher, S. J., and J. A. Sethian, Fronts propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations, J. of Comp. Phys. **79** (1988), 12–49.
17. Rouy, E., and A. Tourin, A viscosity solutions approach to shape-from-shading, SIAM. J. Numer. Analy. **29**(3) (1992), 867–884.
18. Sedgewick, R., *Algorithms*, Addison-Wesley, 1988.
19. Sethian, J. A., A review of the theory, algorithms, and applications of level set methods for propagating interfaces, Acta Numerica, Cambridge University Press, 1996.
20. Sethian, J. A., *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*, Cambridge Univ. Press, 1996.
21. Sethian, J. A., Curvature and the evolution of fronts, Comm. in Math. Phys. **101** (1985), 487–499.
22. Sethian, J. A. A marching level set method for monotonically advancing fronts, Proc. Nat. Acad. Sci. **93**(4), 1996.

Ron Kimmel
Computer Science Department
Technion, Israel Institute of Technology
Haifa 32000, Israel
ron@cs.technion.ac.il
http://www.cs.technion.ac.il/~ron/

James A. Sethian
Department of Mathematics
and Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720, USA
sethian@math.berkeley.edu

On the Geometry of Texture

Ron Kimmel, Nir A. Sochen, and Ravi Malladi

Abstract. We consider texture images as a composition of manifolds in the feature-space. This geometrical interpretation leads to a natural way for texture enhancement. A flow, based on manifold volume minimization, yields a natural enhancement procedure for texture images. The 2D Gabor-Morlet transform is first used to decompose the image into sub-band images, where each sub-image corresponds to a different scale. Each sub-band image may be considered as a 3D manifold in a 5D space from which the original image can be reconstructed in a numerically stable way. Following our previous results, we then invoke Polyakov action from String Theory, and develop a minimization process through a geometric flow that efficiently enhances each sub-band image in a spatial-orientation feature space. Finally, the enhanced sub-band images are composed back into an enhanced texture image.

§1. Introduction

Texture plays an important role in the understanding process of many images. Therefore, it became an important research subject in the fields of psychophysics and computer vision. The study of texture starts from the pre-image that describes the physics and optics that transforms the 3D world into an image, through human perception that starts from the image formation on the retina and tracks its interpretation at the first perception steps in the brain.

The psychophysical research of these first steps focuses on the way the brain cells are activated under the stimulus of a given image. Such experiments combined with recent developments in the field of signal representation led to relatively simple mathematical models that simulate the first steps in the way our brain represents images. One such model is based on the 2D Gabor/Morlet-wavelet transform of the image. Some nice mathematical properties and the relation of this transform to the physiological behavior were studied in [6,10]. This model was used for the segmentation, interpretation and analysis of texture [2,7], for texture based browsing [8], etc.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 203–212.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

In this paper we use the same space to represent texture images. Then, we search for a geometrical way to improve and enhance texture based images. The geometrical feature enhancement procedure we introduce may serve as a step towards segmentation. This procedure is based on a flow in the transformed space in which the transform coefficients are treated as higher dimensional manifolds. A special minimization process preserves domains of constant/homogeneous texture, enhances the texture in each domain, and thereby sharpens the boundaries between neighboring domains with different textures.

The remainder of this paper is organized as follows: Section 2 briefly reviews our previous results: the definition of arclength, the consideration of images as surfaces, and the minimization of Polyakov action that leads to a geometric flow we named the *Beltrami flow.* Next, Section 3 describes the relevant feature space to the texture case. It gives the basics for constructing the 2D Gabor-Morlet wavelet decomposition, and a simple way for composing the image back. Section 4 presents experimental results of the Beltrami flow in the decomposition feature space, for simple gray level texture.

§2. Images as Embedded Maps that Flow Toward Harmonic Maps

In [11] we consider images as 2D surfaces in higher dimensional spaces. We construct enhancement and segmentation procedures for color images as 2D surfaces in 5D $(x, y, \mathbf{r}, \mathbf{g}, \mathbf{b})$ space. As shown in [4], the idea of images as curved spaces is not limited to 2D surfaces, so that movies and volumetric images can be considered as 3D hypersurfaces (manifolds) in 4D $(x, y, z, I(x, y, z))$ space.

Our geometric framework finds a seamless link between the L_1 norm, used in the Osher-Rudin TV image enhancement and its variants, and the L_2 norms, used in Mumford-Shah image segmentation and its variants. TV (Total Variation) schemes are based on minimizing the L_1 norm, namely $\int |\nabla I|$, while the L_2 norm minimizes $\int |\nabla I|^2$. Our framework is based on the geometry of the image and its interpretation as a surface. The aspect ratio between the gray level and the xy image plane, is the switch between the two commonly used norms. This observation made it possible to show that our multi-channel (color) enhancement procedure may be considered as a generalization of the powerful TV scheme that is now commonly used in the high tech image processing industry. This procedure yield very promising results for color image enhancement [11]. In this work, we propose a flow in a rich feature space which is different from the image spatial-intensity space.

Representation and Riemannian structure

We represent an image and other local features as embedding maps of a Riemannian manifold in a higher dimensional space. The simplest example is the image itself which is represented as a 2D surface embedded in $\mathbb{R}^3$. We denote the map by $X : \Sigma \to \mathbb{R}^3$, where Σ is a two-dimensional surface, and we denote the local coordinates on it by (σ^1, σ^2). The map X is given in general by $(X^1(\sigma^1, \sigma^2), X^2(\sigma^1, \sigma^2), X^3(\sigma^1, \sigma^2))$. In our example we represent it as

$(X^1 = \sigma^1, X^2 = \sigma^2, X^3 = I(\sigma^1, \sigma^2))$. We choose a Riemannian structure on this surface, namely, a metric. The metric is a positive definite and symmetric 2-tensor that may be defined through the local distance measurements:

$$ds^2 = g_{\mu\nu} d\sigma^\mu d\sigma^\nu \equiv g_{11}(d\sigma^1)^2 + 2g_{12} d\sigma^1 d\sigma^2 + g_{22}(d\sigma^2)^2, \tag{1}$$

where we used Einstein summation convention in the second equality. We denote the inverse of the metric by $g^{\mu\nu}$.

Polyakov action: a measure on the space of embedding maps

Let us briefly review our general framework for non-linear diffusion in computer vision. We will use this framework in Section 4 to diffuse a textured image in the transformed domain. The equations will be derived by a minimization problem from an action functional. The functional in question depends on *both* the image manifold and the embedding space. Denote by (Σ, g) the image manifold and its metric, and by (M, h) the space-feature manifold and its metric. Then the functional $S[X]$ attaches a real number to a map $X : \Sigma \to M$:

$$S[X^i, g_{\mu\nu}, h_{ij}] = \int dV \langle \nabla X^i, \nabla X^j \rangle_g h_{ij}, \tag{2}$$

where dV is a volume element and $\langle \nabla R, \nabla G \rangle_g = g^{\mu\nu} \partial_\mu R \partial_\nu G$. This functional, for $m = 2$, was first proposed by Polyakov [9] in the context of high energy physics, and the theory is known as string theory.

Using standard methods in the calculus of variations (see [11]), the Euler-Lagrange equations with respect to the embedding are

$$-\frac{1}{2\sqrt{g}} h^{il} \frac{\delta S}{\delta X^l} = \frac{1}{\sqrt{g}} \partial_\mu (\sqrt{g} g^{\mu\nu} \partial_\nu X^i). \tag{3}$$

Since $(g_{\mu\nu})$ is positive definite, $g \equiv \det(g_{\mu\nu}) > 0$ for all σ^μ. This factor is the simplest one that doesn't change the minimization solution while giving a reparameterization-invariant expression. The operator that is acting on X^i is the natural generalization of the Laplacian from flat spaces to manifolds and is called the second order differential parameter of Beltrami [5], or for short Beltrami operator, and we will denote it by Δ_g.

For a surface Σ embedded in 3 dimensional Euclidean space, we get a minimal surface as the solution to the minimization problem. In order to see this and to connect to the usual representation of the minimal surface equation, we notice that the solution of the minimization problem with respect to the metric is

$$g_{\mu\nu} = \partial_\mu X^i \partial_\nu X_i. \tag{4}$$

Plugging this induced metric in the first Euler-Lagrange equation (3), we get the steepest decent flow

$$\mathbf{X}_t = H\mathbf{N}, \tag{5}$$

where H is the mean curvature, and $\mathbf{N}$ is the normal to the surface given by

$$\begin{aligned} H &= \frac{(1+I_x^2)I_{yy} - 2I_xI_yI_{xy} + (1+I_y^2)I_{xx}}{g^{\frac{3}{2}}}, \\ \mathbf{N} &= \frac{1}{\sqrt{g}}(-I_y, -I_x, 1)^T, \end{aligned} \tag{6}$$

and $g = 1 + I_x^2 + I_y^2$. We see that this choice gives us the mean curvature flow! This should not be a surprise, since the action functional for the above choice of metric $g_{\mu\nu}$ is

$$S = \int d\text{Area} = \int d^2\sigma\sqrt{g} = \int d^2\sigma\sqrt{\det(\partial_\mu X^i \partial_\nu X_i)}, \tag{7}$$

which is the Euler functional that describes the area of the surface, also known in high energy physics as the Nambu action.

In general, for any manifold Σ and M, the map $X : \Sigma \to M$ that minimizes the action S with respect to the embedding is called a harmonic map. The harmonic map is the natural generalization of the geodesic curve and the minimal surface to higher dimensional manifolds.

§3. Gabor/Morlet-wavelets: A Natural Space for Texture Images

In [6] Lee argues that the 2D Gabor/Morlet wavelet transform with specific coefficients is an appropriate mathematical description for images. He based his findings on neurophysiological evidence based on experiments on the visual cortex of mammalian brains. These experiments indicate that the best model for the filter response of simple cells are self-similar 2D Gabor/Morlet wavelets.

Following Lee [6], let us briefly describe the 2D Gabor/Morlet wavelets that model the simple cells. The 2D wavelet transform on an image $I(x,y)$ is defined as

$$(T^{wav}I)(x_0, y_0, \theta, a) = \|a\|^{-1} \int\int dxdy I(x,y)\psi_\theta\left(\frac{x-x_0}{a}, \frac{y-y_0}{a}\right), \tag{8}$$

where a is a dilation parameter, x_0 and y_0 are the spatial translations, and θ is the wavelet orientation parameter. Here

$$\psi(x, y, x_0, y_0, \theta, a) = \|a\|^{-1}\psi_\theta(\frac{x-x_0}{a}, \frac{y-y_0}{a}), \tag{9}$$

is the 2D elementary wavelet function rotated by θ. Based on neurophysiological experiments, a specific Gabor elementary function is used as the mother wavelet to generate the 2D Gabor/Morlet wavelet family by convolving the image with

$$\psi(x,y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{1}{8}(4x^2+y^2)}(e^{ikx} - e^{-\frac{k^2}{2}}), \tag{10}$$

and $\psi_\theta(x,y) = \psi(\tilde{x},\tilde{y})$ is defined by rotation of (x,y) via

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}. \tag{11}$$

The discretization of (8), i.e $(T^{wav}_{p,q,l,m} I)$, is denoted by $W_{p,q,l,m}$ and is given by

$$W_{p,q,l,m} = a^{-m} \int\int dxdy I(x,y) \psi_{l\Delta\theta}(a^{-m}(x - p\Delta x), a^{-m}(y - q\Delta x)), \tag{12}$$

where Δx is the basic sampling interval, $\Delta\theta = 2\pi/L$, and the angles are given by $l\Delta\theta$, where $l = 0, ..., L-1$, and L is the total number of orientations. p, q and m are integers determining the position and scaling. Note that as m increases, the sample intervals get larger forming a pyramidal structure. Equation (12) can be read as a projection onto a discrete set of basis functions

$$W_{p,q,l,m} = \langle I, \psi_{p,q,l,m} \rangle. \tag{13}$$

The real number k determines the frequency bandwidth of the filters in octaves via the approximation

$$k = \frac{a^\phi + 1}{a^\phi - 1} \sqrt{2 \ln 2}, \tag{14}$$

where ϕ is the bandwidth in octaves, e.g. for $a = 2$ and $\phi = 1.5$ we get $k \approx 2.5$. In the above approximation the DC normalization term $e^{-k^2/2}$ that is required to make a wavelet basis out of the Gabor basis is ignored, and we consider $a = k/\omega_0$. So the peaks of the scaled mother wavelets in the frequency domain are (approximately) at the locations $a^{-m}\omega_0$.

For our application we have chosen to work with a frame. The concept of frames was introduced in [3]. A family of functions (ψ_j) is a frame if there exist $A > 0, B < \infty$ that are called frame bounds so that for every f we have

$$A\|f\|^2 \leq \sum_j |\langle f, \psi_j \rangle|^2 \leq B\|f\|^2, \tag{15}$$

where $\|f\| = \int f^2$. One could recognize this as a generalization of Parseval's theorem. A discrete family of wavelets that forms a frame provides a complete representation of any function. In some cases it is possible to recover a function with good approximation by the inversion formula

$$f \approx \frac{2}{A+B} \sum_j \langle \psi_j, f \rangle \psi_j. \tag{16}$$

The ratio B/A measures the tightness of the frame. When $A = B$, the frame is tight and the reconstruction by summation is exact. Thus, as B/A approaches 1, we may still use the above reconstruction equation as a good

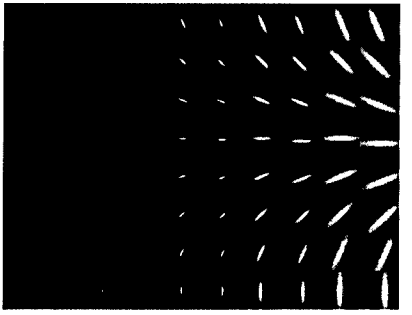


Fig. 1. The wavelet basis functions (up to translations). The basis functions are presented in a gray level array, real (symmetric) and imaginary (a-symmetric) for the 8 angles $[0, \pi]$ and 5 scales.

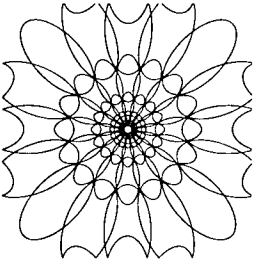


Fig. 2. The half peak contours in the frequency domain of the wavelet basis functions in the previous figure, (5 scales 16 orientations).

approximation. That is, we treat our discrete wavelets as an orthonormal basis.

We denote the 2D Gabor/Morlet-wavelet transform as $W(x, y, \theta, \sigma)$, such that $R = \text{Real}(W)$ and $J = \text{Imag}(W)$, where for the discrete case $\sigma = a^m$ and $\theta = l\Delta\theta$. The response of a simple cell is then modeled by the projection of the image onto a specific Gabor/Morlet wavelet.

Motivated by the arrangement of simple cells in our brain, with as tight a frame as possible, we consider 5 spatial frequency octaves, and 16 angles that discretize the $[0, 2\pi]$ angular interval. Practically, we used the symmetry properties of the 2D Gabor/Morlet-wavelet transform: $W(x, y, \theta + \pi, \sigma) = \bar{W}(x, y, \theta, \sigma)$. Thus, only 8 angles are needed to represent the discretization of the full $[0, 2\pi]$ angular interval into 16. We choose $a = 2$ and $\Delta x = 1$. This selection results in a frame bounds $A = 271.95$, $B = 233.69$, with ratio of $B/A = 1.19$. The fact that this ratio is close to 1 means that we have a tight frame that allows simple summation reconstruction. Figs. 1 and 2 show the basis functions we used. Periodic boundary conditions are used for the real (symmetric) part, and negative periodicity for the imaginary part, forming a 'Klein bottle' coordinate system in (x, y, θ). This enables us to reduce the memory complexity by a factor of 2.

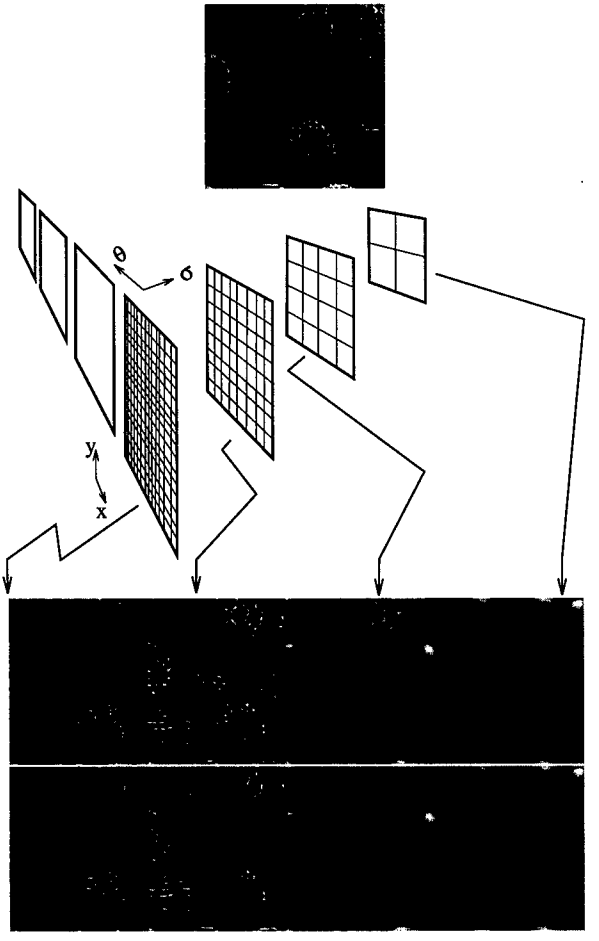


Fig. 3. A schematic diagram of Gabor/Morlet wavelet decomposition of the original image (at the top) into the $(x, y, \theta, W_\sigma(x, y, \theta))$ and the images that are the result of reconstruction by summation for each scale σ separately (bottom). The last row presents the reconstruction result after 70 iteration of the Beltrami flow at each scale. In all the examples we use $L = 16$, $a = 2$, $k = 2.5$, and $m \in \{0, .., 4\}$.

For practical implementation that avoids the special numerical treatment needed along the pyramidal discrete σ scale axis, we consider each scale as a separate space. The induced metric for each scale is then given by

$$(g_{\mu\nu}) = \begin{pmatrix} 1 + R_x^2 + J_x^2 & R_x R_y + J_x J_y & R_x R_\theta + J_x J_\theta \\ R_x R_y + J_x J_y & 1 + R_y^2 + J_y^2 & R_y R_\theta + J_y J_\theta \\ R_x R_\theta + J_x J_\theta & R_y R_\theta + J_y J_\theta & 1 + R_\theta^2 + J_\theta^2 \end{pmatrix}. \qquad (17)$$

This result can be understood from the arclength definition in this spatial-orientation complex space, namely

$$ds^2 = dx^2 + dy^2 + d\theta^2 + dJ^2 + dR^2. \tag{18}$$

Applying the chain rule on $dR = R_x dx + R_y dy + R_\theta d\theta$, and similarly for dJ, we obtain the desired bilinear structure that describes the above induced metric for this case.

The gradient descent equations for the Polyakov action read

$$R_t = \Delta_g R, \qquad J_t = \Delta_g J, \tag{19}$$

where $\Delta_g X$ is given in (3) with the metric (17).

§4. Experimental Results

Let us start with a simple example. In Fig. 3 we first decompose an image via the wavelet transform into 4 separate sub-scale channels. The decomposition and the result of applying the Beltrami flow on each sub-scale are shown.

Let us gain more motivation on the advantage of the wavelet decomposition. Fig. 4 shows the result of composing the image back from just the first 2, and then the first 3 sub-scale channels. The cancellation of the shadowing can also be realized by a very simple high pass filter. However, as a byproduct of the wavelet decomposition, at each scale σ we now have the complex function $W_\sigma(x, y, \theta)$. It defines a surface in the 5D space (3 real and one complex dimensions) (x, y, θ, W_σ). The extra coordinate θ that describes the behavior of the image along a specific direction enables us to smooth the image while keeping the meaningful orientation structure of the texture. Moreover, we have the freedom to apply different filters to the different scales. This enables us to preserve the nature of texture images by processing them only at significant scales. In other words, we are able to sharpen a specific scale without effecting the rest of the sub-band images. Fig. 5 is the original image and the result of applying the Beltrami flow to filter out non-oriented structures. More examples are shown in Fig. 6.

§5. Concluding Remarks

We proposed to combine a psychophysically supported texture space, the 2D Gabor/Morlet-wavelet transform, with a geometrical flow to enhance texture images. The texture was considered as a manifold in its natural space. The flow was realized by invoking Polyakov action, and the result was the Beltrami flow in the feature space. The result is a variational-geometric technique that enhances texture images in their appropriate decomposition space.

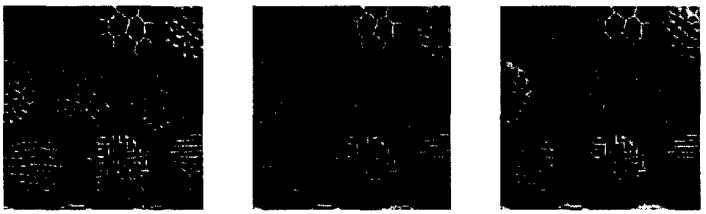


Fig. 4. Reconstruction by summation, of only 2, 3, and all layers of the different scales: the low frequency scale contribute the shadowing, thus summing only over the first 3 scales cancels this effect (a simple high pass effect).

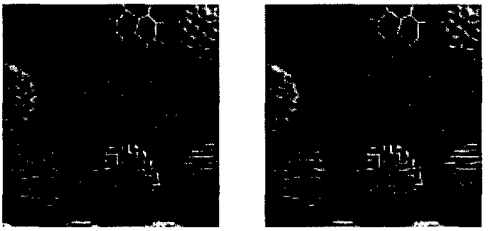


Fig. 5. Left: Original image 128 × 128, Right: Result of Beltrami flow for 70 numerical iterations in each sub-scale.

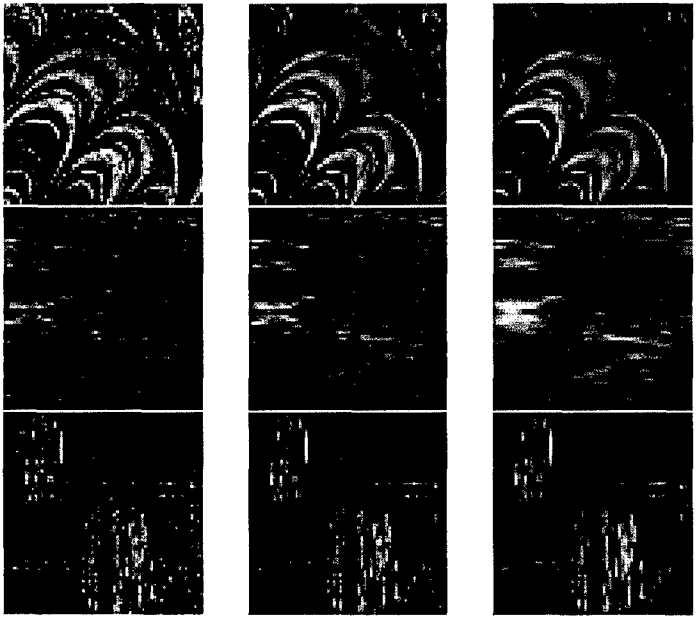


Fig. 6. Example of 2 snapshots from the evolution for different texture images Left: Original image 64 × 64.

References

1. Antoine, J. P., P. Carrette, R. Murenzi and B. Piette, Image analysis with 2-D continuous wavelet transform, Signal Processing **31**(3) (1993), 241–272.
2. Bovik, A. C., M. Clark, and W. S. Geisler, Multichannel texture analysis using localized spatial filters, IEEE Trans. Pattern Anal. and Machine Intelligence **12**(1) (1990), 55–73.
3. Duffin, R. J. and A. C. Schaeffer, A class of nonharmonic Fourier series, Trans. Am. Math. Soc. **72** (1952), 341–366.
4. Kimmel, R., N. Sochen, and R. Malladi, Images as embedding maps and minimal surfaces: Movies, color, and volumetric medical images, *IEEE CVPR'97*, Puerto Rico, June, 1997, 350–355.
5. Kreyszing, E., *Differential Geometry*, Dover Publications, Inc., New York, 1991.
6. Lee, T. S., Image representation using 2D Gabor-wavelets, IEEE Trans. on PAMI **18**(10) (1996), 959–971.
7. Lee, T., D. Mumford, and A. L. Yuille, Texture segmentation by minimizing vector valued energy functionals: the couple-membrane model, in Lecture Notes in Computer Science, 588: ECCV'92, G Sandini (ed.), Springer-Verlag, 1992, 165–173.
8. Manjunath, B. S., and W. Y. Ma, Texture features for browsing and retrieval of image data, IEEE Trans. Pattern Anal. and Machine Intelligence **18**(8) (1996), 837–841.
9. Polyakov, A. M., Physics Letters **103B** (1981), 207.
10. Porat, M. and Y. Y. Zeevi, The generalized Gabor scheme of image representation in biological and machine vision, IEEE Trans. on PAMI **10**(4) (1988), 452–468.
11. Sochen, N., R. Kimmel, and R. Malladi, From high energy physics to low level vision, Report LBNL 39243, LBNL, UC Berkeley, CA 94720, Aug. 1996.

Ron Kimmel, Computer Science Department
Technion, Israel Institute of Technology, Haifa 32000, Israel
ron@cs.technion.ac.il

Nir A. Sochen, School of Mathematical Sciences
Tel-Aviv University, Tel-Aviv 69978, Israel
sochen@math.tau.ac.il

Ravi Malladi, Lawrence Berkeley National Laboratory
University of California, Berkeley, CA 94720, USA
malladi@math.lbl.gov

Optimization of a Curve Approximation Based on NURBS interpolation

Jérôme Lépine, François Guibault,
Marie-Gabrielle Vallet, and Jean-Yves Trépanier

Abstract. In this paper, an approach is presented whereby optimal spatial positions and weights of a fixed number of NURBS control points are determined using a quasi-Newton optimization algorithm in order to approximate a general planar target curve. A method for constructing an adequate initial solution and a valid cost function based on interpolation error are introduced. Convergence of the iterative process is assessed, and the final interpolation error is related to prescribed manufacturing or analysis tolerances. The efficiency of the approach is demonstrated for actual wing profiles.

§1. Introduction

The problem of constructing a cost effective approximation of a general target curve is of great relevance in many engineering disciplines. This problem has been addressed quite thoroughly in the context of polynomial interpolation [1], but far less work has been published on rational approximation. Indeed, weights introduce another level of difficulty in the theoretical analysis of the approximation error. From a practical standpoint though, non-uniform rational B-Splines [4] (NURBS) provide more degrees of freedom for a given number of control points, which leads naturally to smoother curves.

The work presented here introduces a robust numerical approach for the determination of control point positions and weights of a NURBS curve; it can be used to construct an approximation to a general target planar curve. In the context of wing profile design, where this approach has been applied [2,3], very significant reductions in terms of data size and noise level have been observed.

In this paper, the approximation problem is first presented, and the method of computation of the approximation error discussed. Next, the optimization method itself is presented, including the choice of initial solutions. Finally, the performance of the method is evaluated for practical test cases, and conclusions are drawn.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 213–220.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

§2. Approximation Problem

A NURBS curve is defined such that

$$\boldsymbol{A}(u) = \sum_{i=0}^{n} R_{i,p}(u)\boldsymbol{P}_i \tag{1}$$

with

$$R_{i,p}(u) = \frac{N_{i,p}(u)\omega_i}{\sum_{j=0}^{n} N_{j,p}(u)\omega_j}, \tag{2}$$

where $\boldsymbol{P}_i$ are the control point coordinates, ω_i their respective weights, $N_{i,p}$ the p-th degree B-spline basis functions and $\boldsymbol{A}(u)$ the position of a point on the curve. The basis functions are obtained through a *knot vector*, which defines the functions' break points, of the form

$$\{\underbrace{0,\ldots,0}_{p}, u_{p+1},\ldots,u_{m-p+1}, \underbrace{1,\ldots,1}_{p}\}.$$

Using these interpolation functions, the problem of approximating a general planar curve $\mathcal{C}(t)$ can be stated as follows: *find the set of control points* $\boldsymbol{P}_i$ *and weights* ω_i *such that* $\| \boldsymbol{A}(u) - \mathcal{C}(t) \|$ *is minimized in a suitable norm.*

Analytically, the L_2 norm would be a natural choice; numerically, though, for a completely general taget curve, this norm can only be approximated through discretization. Numerical experiments have thus been carried out to develop and validate a robust computational approach for the determination of the appoximation error. Consideration has been given to both the mean and maximum error, as well as to the level of continuity of the target curve. Three classes of target curves have been considered: curves only given as a set of points, piecewise linear curves, and C^1 or more continuous curves. In all cases, the mean error

$$\varepsilon_{mea} = \frac{1}{n}\sum_{k=1}^{n} d_k \tag{3}$$

is determined by summing the distance (d_k) of a set of points chosen on the target curve to their respective projection on the approximation curve, and the maximum error

$$\varepsilon_{max} = max_{1\leq k\leq n} d_k \tag{4}$$

gives the largest of these distances.

As a sample of these experiments, Fig. 1 illustrates the behaviour of the approximation error between two typical target curves, a piecewise linear (a) and a quadratically interpolated B-Spline curve (b), and their approximation constructed using a NURBS curve with 13 control points. Both target curves were specified using 143 control points. As can be observed from the graphs of Fig. 2, a very large number of evenly spaced discretisation points must be used to accurately compute both the mean and maximum approximation errors for the piecewise linear case. The behaviour for quadratic test cases and higher

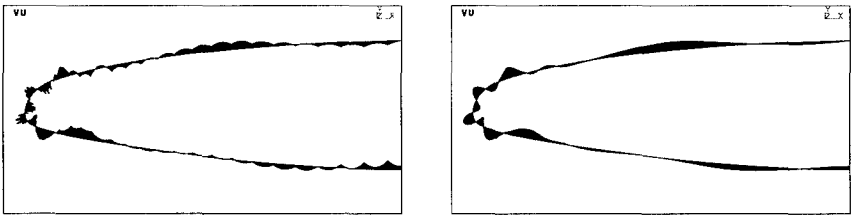


Fig. 1. Approximation error for piecewise linear (left) and quadratic (right) curve (magnified 100X).

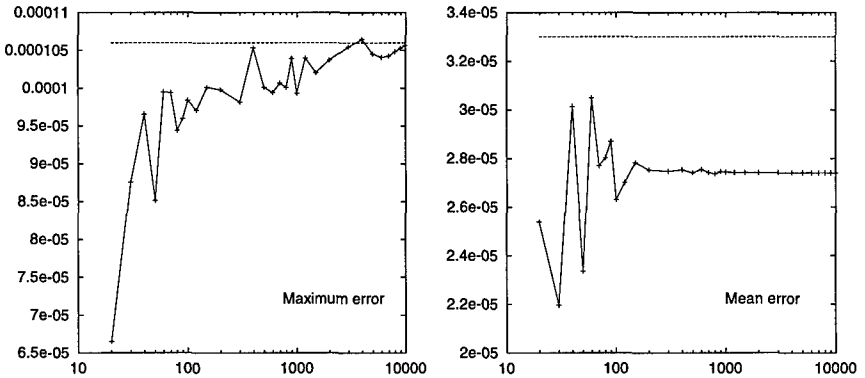


Fig. 2. Maximum (left) and mean(right) error as a function of the number of integration points for the piecewise linear target.

degree of continuity examples (not shown) are extremely similar. The same graphs of Fig. 2 also show, as a straight line, the error computed using only the definition control points of the target curve.

In light of these experiments, it was determined that the error computed using the control points constitutes an adequate bound on both the mean and maximum error of approximation, and it can be computed at a fraction of the cost of using evenly spaced discretization points. This method of computing the error also has the property that it includes naturally the case of target curves given as a discrete set of points, which is not a rare case in many practical applications.

§3. Optimization Method

Using these definitions and computational method of the approximation error, the optimization problem can be further specified by introducing a cost function of the form

$$F(\mathbf{X}) = 2 \times \varepsilon_{mea} + \varepsilon_{max},$$

where $\boldsymbol{X}$ is the vector of design variables, in this case the position and weights of the approximation curve: $\boldsymbol{X} = \{x_1, y_1, \omega_1, x_2, \ldots, x_n, y_n, \omega_n\}$. This choice

of a cost function significantly accelerates convergence of the optimization process by including both the maximum error, which controls the quality of the final approximation, and the mean error, which globally compares the quality of different solutions.

Clearly, this is a non-linear optimization problem, and we will now examine the chosen solution process, including the choice of an initial solution.

Solution method

The primary solution method used was the second-order quasi-Newton method, which, given a reasonably close initial solution $\boldsymbol{X_0}$, will iteratively converge towards an optimal solution using the relation

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \alpha_k \, \boldsymbol{S}_k,$$

where $\boldsymbol{S}_k = -\boldsymbol{H}_k \cdot \nabla F(\boldsymbol{X}_k)$ is the direction of descent vector, and α_k the distance of descent in direction $\boldsymbol{S}_k$. The descent vector is computed using the BFGS [6] algorithm, based on a second order approximation of the gradient of $F(\boldsymbol{X})$:

$$\nabla F(\boldsymbol{X}) \simeq \nabla F(\boldsymbol{X}_k) + \boldsymbol{H}(\boldsymbol{X}_k) \cdot \delta \boldsymbol{X},$$

where $\delta \boldsymbol{X} = \boldsymbol{X} - \boldsymbol{X}_k$ is used as the direction of descent vector $(\boldsymbol{S}_k)$. Here $\boldsymbol{H}$, the approximate Hessian matrix, is initially set to identity and iteratively updated using the relation

$$\boldsymbol{H}_{k+1} = \boldsymbol{H}_k + \frac{\boldsymbol{Y}_k \otimes \boldsymbol{Y}_k}{\boldsymbol{Y}_k \cdot \boldsymbol{S}_k} - \frac{(\boldsymbol{H}_k \cdot \boldsymbol{S}_k) \otimes (\boldsymbol{H}_k \cdot \boldsymbol{S}_k)}{\boldsymbol{S}_k \cdot \boldsymbol{H}_k \cdot \boldsymbol{S}_k}$$

with $\boldsymbol{Y}_k = \nabla F(\boldsymbol{X}_{k+1}) - \nabla F(\boldsymbol{X}_k)$. The distance of descent is computed using Armijo's rule [5], where $\alpha_k = (\frac{1}{2})^m$ and m is the smallest integer such that the relationship

$$F(\boldsymbol{X}_k + \alpha_k \, \boldsymbol{S}_k) \leq F(\boldsymbol{X}_k) + \sigma \alpha_k \nabla F(\boldsymbol{X}_k) \cdot \boldsymbol{S}_k$$

with σ the sufficient descent criterion, which must be chosen between 0 and $\frac{1}{2}$ (usually set to 10^{-4}).

Initial solution

In most cases, the optimization method described above will find a solution, but in the case of highly non-linear cost functions such as the one used in this problem, it is impossible to determine whether the minimum found is the global minimum or only a local one. The only way to circumvent this difficulty is to proceed with many optimizations, and select the best minimum as the solution. While this approach could be unaffordable if no clue were available about the solution, it can be implemented relatively cheaply in the context of curve approximation, where many good initial guesses can be constructed.

Specifically, a set of initial solutions is constructed by discretizing the target curve using a fixed number of points and by varying the concentration of

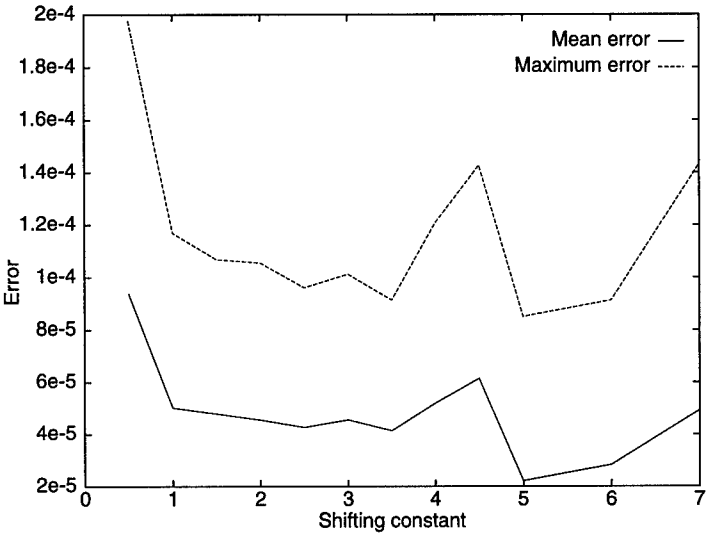


Fig. 3. Optimal Approximation error for various shifting constant values.

points along the curve. Basically, points are gathered closer together in regions of high curvature, and a shifting constant is introduced to construct various concentration laws. For target curves of continuity less than C^2, curvature is approximated using centered differencing. The concentration law is evaluated using

$$\mathcal{F}(u) = \frac{1}{K} \int_0^1 C(v) + D \, dv,$$

where $C(v)$ is the true or approximated curvature of the target curve, and D the shifting constant.

When the shifting constant becomes large, the concentration law becomes almost uniform. In practice, sets of 8 to 10 initial solutions are constructed by varying D typically between 1.0 and 10, and each initial solution is then optimized. Fig. 3 shows the final approximation error for a run where D took the values $\{0.5, 1.0, 2.5, 3.0, 3.5, 4.5, 5.0, 6.0, 7.0\}$. The target curve for this problem is a standard NACA 2412 wing profile, and 9 control points are used for the approximation, which leads to a 21 parameter optimization problem (the two endpoints are fixed). Initial weights are all set to 1.0.

Fig. 3 vividly illustrates the high non-linearity of the problem, where small variations in the initial solution lead to completely different optimal solutions, as expressed, for example, by the steep variation in final error for $D = 4.5$ and $D = 5.0$.

§4. Application

We will now look at how this approximation method performs in the context of a practical application, both in terms of data reduction and approximation

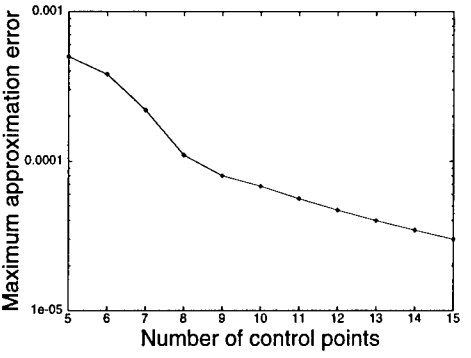


Fig. 4. Precision of the approximation as the number of control points increases.

characteristics. Two aspects of the approximation are of particular interest: precision and noise level.

The application consists in approximating wing profiles, specified either as analytical functions or experimental sets of points. In this context, study [3] of the combined precision levels dictated by both manufacturing tolerances and precision for analysis purposes indicates that a precision of the order of 8×10^{-5} is sufficient.

Precision

Fig. 4 illustrates the evolution of ε_{max} as the number of control points of the approximation curve is increased. Again the target curve is the NACA 2412 profile. As can be observed, the increase in precision of the approximation is very regular when 8 control points or more are used. In this case, the required precision of 8×10^{-5} is obtained with only 9 control points. Extensive experiments [3] involving numerous types of wing profiles have shown that the required level of precision can almost always be attained with 13 control points or less. These numbers have to be compared with the number of points needed to discretely represent a profile with the same precision, which can be shown to be of the order of 150. The approximation method thus offers excellent control over the precision of the resulting curve, while reducing by more than an order of magnitude the amount of data used for representation.

Experiments have also been carried out in order to determine whether the introduction of weights in the formulation had an impact on precision. Similar precision tests carried out using B-Splines instead of NURBS have shown that exactly 1.5 more control points were required for B-Splines to obtain comparable levels of precision. This increase in the number of control points, however, has a significant impact on noise level.

Noise

For many engineering applications, such as airplane wing design, noise level is often a bigger concern than absolute precision level. In that respect, the NURBS approximation method performs remarkably well, mainly because

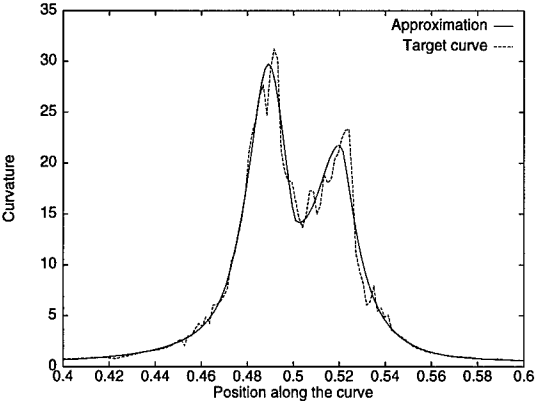


Fig. 5. Curvature of the Boeing A8 profile and of its approximation near the leading edge.

of the small number of control points needed. Noise appears as small fluctuations in the curvature of a curve, particularly when control points are gathered closely together in regions of high curvature. As shown in Fig. 5, the proposed approximation method can significantly reduce noise in cases where the target curve presents important fluctuations. Of course, this reduction of the noise level can only be accomplished as a trade-off to the precision of the approximation. For example, the precision of the 13 control point approximation of the Boeing A8 profile of Fig. 5 is 9.2×10^{-5}, which is slightly above the usual tolerance level for this application; better precision could be obtained by including a few more control points, but this would inevitably introduce more noise.

§5. Conclusion

We have presented a method of approximation for a general planar curve that permits a significant reduction in the size of data and garantees a desired level of precision. The main advantages of this approach are

- generality,
- full automation,
- low noise.

By varying a single parameter, the shifting constant in the construction of the concentration law, as many initial solutions as needed are generated for a given number of points. Each solution is optimized independently, and the solution with minimum error is kept. Because of the typically small number of control points required – of the order of 10 to 15 – very smooth curves are obtained, which is a very important characteristic for many engineering applications.

Because of the significant reduction in the number of free parameters used to represent a curve, the approximation method is now being used as

a first step in a shape optimization procedure of wing profiles. This work is also currently being extended to three dimensional cases, where the method is now used to approximate wing surfaces.

References

1. Hoschek, J. and N. Wissel, Optimal approximate conversion of spline curves and spline approximation of offset curves, Computer-Aided Design **20**, 475-483.
2. Lépine, J., Trépanier, J.-Y. and F. Pépin, Wing aerodynamic design using an optimized NURBS geometrical representation, 38th AIAA Aerospace Sciences Meeting and Exhibit, January 10-13, 2000.
3. Lépine, J., Optimisation de la représentation de profils d'ailes pour application au design aérodynamique, M.Sc.A. thesis École Polytechnique de Montréal, Avril 1999.
4. Piegl, L. and W. Tiller, *The NURBS Book*, Springer, 1995.
5. Press, W. H., *Numerical Recipes in C : the Art of Scientific Computing, 2 ed.* Cambridge University Press, Cambridge (UK), 1992.
6. Vanderplaats, G. N. *Numerical Optimization Techniques for Engineering Design with Applications.* Series in mechanical engineering. McGraw-Hill, 1984.

Jérôme Lépine
Département de génie mécanique, École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville, Montréal (Québec)
H3C 3A7, Canada
lepine@cerca.umontreal.ca

François Guibault
Département de génie électrique et de génie informatique
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville, Montréal (Québec)
H3C 3A7, Canada
francois@cerca.umontreal.ca

Marie-Gabrielle Vallet
Advanced Aerodynamics
Bombardier Aerospace
Marie-Gabrielle.Vallet@eng.canadair.ca

Jean-Yves Trépanier
Département de génie mécanique, École Polytechnique de Montréal
C.P. 6079, Succ. Centre-Ville, Montréal (Québec)
H3C 3A7, Canada
jyves@cerca.umontreal.ca

Filling N-sided Holes Using Combined Subdivision Schemes

Adi Levin

Abstract. A new method is presented for calculating N-sided surface patches. The algorithm generates a subdivision surface which satisfies arbitrary C^1 boundary conditions. The proposed subdivision scheme is based on a Catmull-Clark type subdivision scheme that operates in the surface interior. Near the boundary we introduce new subdivision rules that involve the given boundary conditions. The generated subdivision surface is C^2-continuous except at one extraordinary point. In the neighborhood of this point the surface curvature is bounded.

§1. Background

The problem of constructing N-sided surface patches occurs frequently in computer-aided geometric design. The N-sided patch is required to connect smoothly to given surfaces surrounding a polygonal hole, as shown in Fig. 1.

Referring to [10,25,26], N-sided patches can be generated basically in two ways. Either the polygonal domain, which is to be mapped into 3D, is subdivided in the parametric plane, or one uniform equation is used to represent the entire patch. In the former case, triangular or rectangular elements are put together [2,6,12,20,23] or recursive subdivision methods are applied [5,8,24]. In the latter case, either the known control-point based methods are generalized, or a weighted sum of 3D interpolants gives the surface equation [1,3,4,22].

The method presented in this paper is a recursive subdivision scheme specially designed to consider arbitrary boundary conditions. Subdivision schemes provide efficient algorithms for the design, representation and processing of smooth surfaces of arbitrary topological type. Their simplicity and their multiresolution structure make them attractive for applications in 3D surface modeling, and in computer graphics [7,9,11,13,19,27,28].

The subdivision scheme presented in this paper falls into the category of combined subdivision schemes [14,15,17,18], where the underlying surface is

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 221–228.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

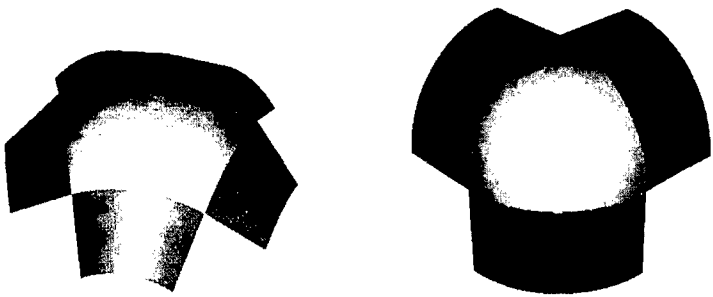


Fig. 1. A 5-sided and a 3-sided surface patch.

represented not only by a control net, but also by the given boundary conditions. The scheme repeatedly applies a subdivision operator to the control net, which becomes more and more dense. In the limit, the vertices of the control net converge to a smooth surface. Samples of the boundary conditions participate in every iteration of the subdivision, and as a result the limit surface satisfies the given conditions, regardless of their representation. Thus, our scheme performs so-called transfinite interpolation.

The motivation behind the specific subdivision rules, and the smoothness analysis of the scheme are presented in [16]. In the following sections, we describe Catmull-Clark's scheme, and we present the details of our scheme.

§2. Catmull-Clark Subdivision

A net $\Sigma = (V, E)$ consists of a set of vertices V and the topological information of the net E, in terms of edges and faces. A net is closed when each edge is shared by exactly two faces.

Camull-Clark's subdivision scheme is defined over closed nets of arbitrary topology, as an extension of the tensor product bi-cubic B-spline subdivision scheme [5,8]. Variants of the original scheme were analyzed by Ball and Storry [24]. Our algorithm employs a variant of Catmull-Clark's scheme due to Sabin [21], which generates limit surfaces that are C^2-continuous everywhere except at a finite number of irregular points. In the neighborhood of those points the surface curvature is bounded. The irregular points come from vertices of the original control net that have valency other than 4, and from faces of the original control net that are not quadrilateral.

Given a net Σ, the vertices V' of the new net $\Sigma' = (V', E')$ are calculated by applying the following rules on Σ (see Fig. 2):

1) For each old face f, make a new face-vertex $v(f)$ as the weighted average of the old vertices of f, with weights W_m that depend on the valency m of each vertex.

2) For each old edge e, make a new edge-vertex $v(e)$ as the weighted average of the old vertices of e and the new face vertices associated with the

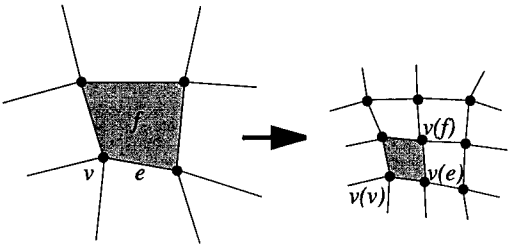


Fig. 2. Catmull-Clark's scheme.

two faces originally sharing e. The weights W_m (which are the same as the weights used in rule 1) depend on the valency m of each vertex.

3) For each old vertex v, make a new vertex-vertex $v(v)$ at the point given by the following linear combination, whose coefficients $\alpha_m, \beta_m, \gamma_m$ depend on the valency m of v:

$\alpha_m \cdot$ (the centroid of the new edge vertices of the edges meeting at v) + $\beta_m \cdot$ (the centroid of the new face vertices of the faces sharing those edges) + $\gamma_m \cdot v$.

The topology E' of the new net is calculated by the following rule: for each old face f and for each vertex v of f, make a new quadrilateral face whose edges join $v(f)$ and $v(v)$ to the edge vertices of the edges of f sharing v (see Fig. 2).

We present the procedure for calculating the weights mentioned above, as formulated by Sabin in [21]: Let $m > 2$ denote a vertex valency. Let $k := \cos(\pi/m)$. Let x be the unique real root of

$$x^3 + (4k^2 - 3)x - 2k = 0,$$

satisfying $x > 1$. Then

$$W_m = x^2 + 2kx - 3, \quad \alpha_m = 1,$$

$$\gamma_m = \frac{kx + 2k^2 - 1}{x^2(kx + 1)}, \quad \beta_m = -\gamma_m.$$

The weights W_m and γ_m for $m = 3, \ldots, 7$ are given by

$$\begin{aligned}
W_3 &= 1.23606797749979\ldots & \gamma_3 &= 0.06524758424985\ldots \\
W_4 &= 1.00000000000000000 & \gamma_4 &= 0.25000000000000000 \\
W_5 &= 0.71850240323974\ldots & \gamma_5 &= 0.40198344690335\ldots \\
W_6 &= 0.52233339335931\ldots & \gamma_6 &= 0.52342327689253\ldots \\
W_7 &= 0.39184256502794\ldots & \gamma_7 &= 0.61703187134796\ldots
\end{aligned}$$

Remark: The original paper by Sabin [21] contains a mistake: the formulas for the parameters α, β and γ that appear in §4 there are $\beta := 1$, $\gamma := -\alpha$.

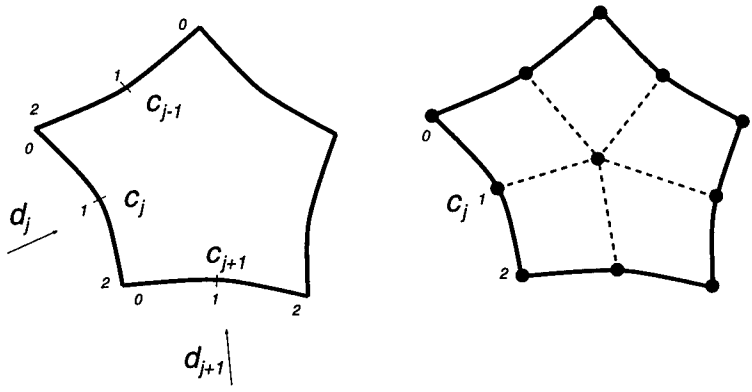


Fig. 3. The input data (left) and the initial control net (right).

§3. The Boundary Conditions

The input to our scheme consists of N smooth curves given in a parametric representation $c_j : [0,2] \to \mathbb{R}^3$ over the parameter interval $[0,2]$, and corresponding cross-boundary derivative functions $d_j : [0,2] \to \mathbb{R}^3$ (see Fig. 3). We say that the boundary conditions are C^0-compatible at the j-th corner if

$$c_j(2) = c_{j+1}(0).$$

We say that the boundary conditions are C^1-compatible if

$$d_j(0) = -c'_{j-1}(2),$$
$$d_j(2) = c'_{j+1}(0).$$

We say that the boundary conditions are C^2-compatible if the curves c_j have Hölder continuous second derivatives, the functions d_j have Hölder continuous derivatives, and the following twist compatibility condition is satisfied:

$$d'_j(2) = -d'_{j+1}(0). \tag{1}$$

The requirement of Hölder continuity is used in [16] for the proof of C^2-continuity in case the boundary conditions are C^2-compatible.

§4. The Algorithm

In this section we describe our algorithm for the design of an N-sided patch satisfying the boundary conditions described in §3. The key ingredients of the algorithm are two formulas for calculating the boundary vertices of the net. These formulas are given in §4.3 and §4.4.

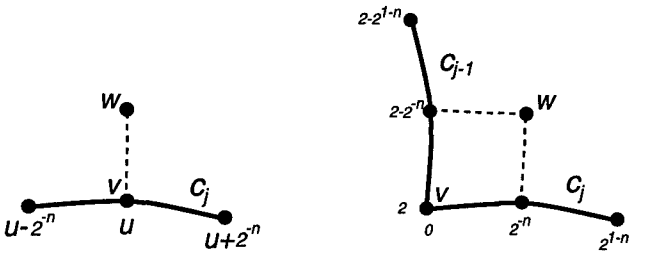


Fig. 4. The stencils for the smooth boundary rule (left) and the corner rule (right).

4.1. Constructing an initial control net

The algorithm starts by constructing an initial control net whose faces are all quadrilateral with $2N$ boundary vertices and one middle vertex, as shown in Fig. 3. The boundary vertices are placed at the parameter values $0, 1, 2$ on the given curves. The middle vertex can be arbitrarily chosen by the designer, and controls the shape of the resulting surface.

4.2. A single iteration of subdivision

We denote by n the iteration number, where $n = 0$ corresponds to the first iteration. In the n-th iteration we perform three steps: First, we relocate the boundary vertices according to the rules given below in §4.3 - §4.4. Then, we apply Sabin's variant of Catmull-Clark's scheme to calculate the new net topology and the position of the new internal vertices. For the purpose of choosing appropriate weights in the averaging process, we consider the boundary vertices as if they all have valency 4. This makes up for the fact that the net is not closed. In the third and final step, we sample the boundary vertices from the given curves at uniformly spaced parameter values with interval length $2^{-(n+1)}$.

4.3. A smooth boundary rule

Let v denote a boundary vertex corresponding to the parameter $0 < u < 2$ on the curve c_j. Let w denote the unique internal vertex which shares an edge with v (see Fig. 4 (left)). In the first step of the n-th iteration we calculate the position of the v by the formula

$$v = 2c_j(u) - \frac{1}{2}w - \frac{1}{4}\left(c_j\left(u+2^{-n}\right) + c_j\left(u-2^{-n}\right)\right) - 2^{-n}\frac{1}{12}\left(d_j\left(u+2^{-n}\right) + d_j\left(u-2^{-n}\right)\right) + 2^{-n}\frac{2}{3}d_j(u).$$

4.4. A corner rule

Let v denote a boundary vertex corresponding to the point $c_{j-1}(2) = c_j(0)$. Let w be the unique internal vertex sharing a face with v (see Fig. 4 (right)). In the first step of the n-th iteration we calculate the position of v by the formula

$$\begin{aligned} v = &\frac{5}{2}c_j(0) + \frac{1}{4}w - \left(c_j(2^{-n}) + c_{j-1}(2-2^{-n})\right) + \frac{1}{8}c_j(2^{1-n}) \\ &+ \frac{1}{8}c_{j-1}(2-2^{1-n}) + 2^{-n}\frac{29}{48}\left(d_j(0) + d_{j-1}(2)\right) - 2^{-n}\frac{1}{12}d_j(2^{-n}) \\ &- 2^{-n}\frac{1}{12}d_{j-1}(2-2^{-n}) - 2^{-n}\frac{1}{48}\left(d_j(2^{1-n}) + d_{j-1}(2-2^{1-n})\right). \end{aligned}$$

§5. Properties of our Scheme

In [16] we prove that the vertices generated by the above procedure converge to a surface which is C^2-continuous almost everywhere, provided that the boundary conditions are C^2-compatible (as defined in §3). The only point where the surface is not C^2-continuous is a middle-point (corresponding to the middle vertex, which has valency N), where the surface is only G^1-continuous. In the neighborhood of this extraordinary point, the surface curvature is bounded.

The limit surface interpolates the given curves, for C^0-compatible boundary conditions. For C^1-compatible boundary conditions, the tangent plane of the limit surface at the point $c_j(u)$ is spanned by the vectors $c_j'(u)$ and $d_j(u)$, thus the surface satisfies C^1-boundary conditions. Furthermore, due to the locality of this scheme, the limit surface is C^2 near the boundaries except at points where the C^2-compatibility condition is not satisfied.

The surfaces in Fig. 5 demonstrate that the limit surface behaves moderately even in the presence of wavy boundary conditions. The limit surfaces are C^2-continuous near the boundary except at corners where the twist compatibility condition (1) is not satisfied.

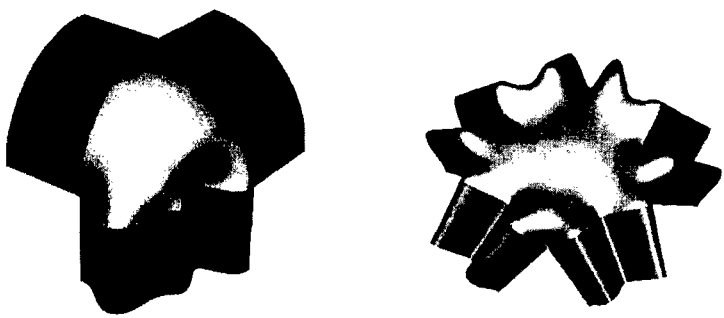


Fig. 5. A 5-sided and a 3-sided surface patch with wavy boundary curves.

Acknowledgments. This work is sponsored by the Israeli Ministry of Science. The author wishes to thank Nira Dyn for her guidance and many helpful comments.

References

1. Barnhill, R. E., Computer aided surface representation and design, in *Surfaces in CAGD*, R. E. Barnhill and W. Boehm (eds), North-Holland, Amsterdam, 1986, 1–24.
2. Becker, E., Smoothing of shapes designed with free-form surfaces, Computer Aided Design **18**(4) (1986), 224–232.
3. Boehm, W., Triangular spline algorithms, Computer Aided Geometric Design **2**(1) (1985), 61–67.
4. Boehm, W., G. Farin, and J. Kahmann, A survey of curves and surface methods in CAGD, Computer Aided Geometric Design **1**(1) (1985), 1–60.
5. Catmull, E., and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, Computer Aided Design **10** (1978), 350–355.
6. Chiokura, H., Localized surface interpolation method for irregular meshes, in *Advanced Computer Graphics*, Proc. Comp. Graphics, L. Kunii (ed.), Tokyo, Springer, Berlin, 1986.
7. DeRose, T., M. Kass, and T. Truong, Subdivision surfaces in character animation, in *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, ACM SIGGRAPH, 1998, 85–94.
8. Doo, D., and M. Sabin, Behaviour of recursive division surface near extraordinary points, Computer Aided Design **10** (1978), 356–360.
9. Dyn, N., J. A. Greogory, and D. Levin, A butterfly subdivision scheme for surface interpolation with tension control, ACM Transactions on Graphics **9** (1990), 160–169.
10. Gregory, J. A., V. K. H. Lau, and J. Zhou, Smooth parametric surfaces and N-sided patches, in *Computation of Curves and Surfaces*, ASI Series, W. Dahmen, M. Gasca, and C. A. Micchelli (eds.), Kluwer Academic Publishers, Dordrecht, 1990, 457–498.
11. Halstead, M., M. Kass, and T. DeRose, Efficient, fair interpolation using catmull-clark surfaces, in *SIGGRAPH 93 Conference Proceedings*, Annual Conference series, ACM SIGGRAPH, 1993, 35–44.
12. Herron, G. J., Triangular and multisided patch schemes, PhD thesis, University of Utah, Salt Lake City, UT, 1979.
13. Kobbelt, L., T. Hesse, H. Prautzsch, and K. Schweizerhof, Interpolatory subdivision on open quadrilateral nets with arbitrary topology, Computer Graphics Forum 15, Eurographics '96 issue, 1996, 409–420.
14. Levin, A., Analysis of combined subdivision schemes 1, Submitted, 1999, Available on the web at the author's home-page.

15. Levin, A., Analysis of combined subdivision schemes 2, In preparation, 1999, Available on the web at the author's home-page.
16. Levin, A., Combined Subdivision Schemes and their applications to 3D modeling. PhD thesis, Tel-Aviv university, 1999, to appear.
17. Levin, A., Combined subdivision schemes for the design of surfaces satisfying boundary conditions, Computer Aided Geometric Design **16**(5) (1999), 345-354.
18. Levin, A., Interpolating nets of curves by smooth subdivision surfaces, *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, 1999, 57–64.
19. Loop, C., Smooth spline surfaces based on triangles, Master's thesis, University of Utah, Department of Mathematics, 1987.
20. Nadler, E., A practical approach to N-sided patches, presented at the Fourth SIAM Conference on Geometric Design, Nashville, 1995.
21. Sabin, M., Cubic recursive division with bounded curvature, in *Curves and Surfaces*, P. J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press, 1991, 411–414.
22. Sabin, M. A., Some negative results in N-sided patches, Computer Aided Design **18**(1) (1986), 38–44.
23. Sarraga, R. F., G^1 interpolation of generally unrestricted cubic Bézier curves, Computer Aided Geometric Design **4** (1987), 23–29.
24. Storry, D. J. T., and A. A. Ball, Design of an N-sided surface patch, Computer Aided Geometric Design **6** (1989), 111–120.
25. Varady, T., Survey and new results in n-sided patch generation, in *The Mathematics of Surfaces II*, R. Martin (ed.), Oxford Univ., 1987, 203–235.
26. Varady, T., Overlap patches: a new scheme for interpolating curve networks with N-sided regions, Computer Aided Geometric Design **8** (1991), 7–27.
27. Zorin, D., P. Schröder, and W. Sweldens, Interpolating subdivision for meshes with arbitrary topology, Computer Graphics Proceedings (SIGGRAPH 96), 1996, 189–192.
28. Zorin, D., P. Schröder, and W. Sweldens, Interactive multiresolution mesh editing, Computer Graphics Proceedings (SIGGRAPH 97), 1997, 259–268.

Adi Levin
Tel-Aviv university, School of Mathematical Sciences,
Tel-Aviv 69978, Israel
adilev@inter.net.il
http://www.math.tau.ac.il/~adilev

Dealing with Topological Singularities in Volumetric Reconstruction

H. Lopes, L. G. Nonato, S. Pesco, and G. Tavares

Abstract. In this work we introduce a new representation for 3-dimensional stratified manifolds based on Morse theory. This representation, which we call *Handle-Strata*, includes a new data structure and a set of operators. Applications of this representation on the volumetric reconstruction from planar sections are presented.

§1. Introduction

Given a set of planar sections of an *object*, by definition a smooth 3-dimensional stratified manifold, the volumetric reconstruction problem consists in building a *geometric model* that is an approximation for this object. In this paper we work with piecewise-linear approximations.

There are several strategies for solving the 3-dimensional reconstruction problem, such as: heuristic, voxel, implicit, parametrical and optimal. Some of these techniques build the surfaces, which are the boundary of the solid object, while others generate a 3-dimensional cell decomposition of the object volume. Two of the main softwares in this area are the 1) Nuages software [8], developed by the PRISME project at INRIA Sophie Antipolis based on surface reconstruction and Volvis [11], and 2) software developed by the VolVis project at the Visualization Lab, Computer Science Department, SUNY at Stone Brook based on voxel reconstruction.

Three problems are intrinsic to the reconstruction process, namely: correspondence, tiling and branching. Correspondence consists in defining the connected components of the model. Tiling means to triangulate the strip between two adjacent slices with respect to some criteria. The branching problem is related to the identification of the object's *singularities*.

Boissonnat introduced an important heuristic technique based on computational geometry concepts of proximity [1]. This technique makes use of the 3-dimensional Delaunay Triangulation and Voronoi Diagram to generate

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 229–238.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

the geometric model. For a definition of Voronoi Diagram and Delaunay Triangulation of a discrete set of different points in $\mathbb{R}^3$ see [3].

The advantages of using the Delaunay triangulation in reconstruction problems are: regions which are geometrically well positioned, with respect to some proximity measure can be found through topological tests; a volumetric triangulation connecting the regions is automatically generated; the volumetric triangulation is appropriated for applications in simulations.

However, without a suitable object representation, the advantages above cannot be fully realized. One of the main reasons is that the representation has to deal with the topological singularities that may appear during the process of reconstruction or even in post-processing applications, e.g., in applying finite element methods to deform the objects.

The main purpose of this work is to introduce a new representation (data structure and its operators) for the cell decomposition of an object. This representation is called *Handle-Strata* (HS-Rep for short). A second goal is to discuss the applications for this new representation in the reconstruction process.

The paper is organized as follows. Section 2 introduces the *Handle-Strata* representation. Section 3 discusses one reconstruction method based on Delaunay Triangulation, and identifies the role of singularities in the reconstruction process. Section 4 shows the applications of this new representation to volumetric reconstruction. Finally, in Section 5 we show images of some reconstructed objects.

§2. Handle-Strata Representation: Data Structure and Operators

In [2], Castelo, Lopes and Tavares introduced a representation for surfaces with boundary based on Morse theory [4]. Lopes and Tavares in [6] extended it to deal with 3-manifolds with boundary. In [9], Pesco devised a representation for stratified surfaces also on Morse theory.

The representation we introduce in this paper is for the 3-dimensional cellular decomposition of an object in $\mathbb{R}^3$. The HS-Rep is an extension of [9] to deal with stratified 3-manifolds. A 3-dimensional cellular decomposition of a subset $\mathcal{K}$ in $\mathbb{R}^3$ is a collection $\mathcal{C}$ of i-dimensional cells ($i = 0, 1, 2, 3$) in $\mathbb{R}^3$ under the following conditions:

1) $\mathcal{K} = \cup\{\sigma \in \mathcal{C}\}$,
2) If σ and $\tau \in \mathcal{C}$ then $\sigma \cap \tau \in \mathcal{C}$, where this intersection is either empty or a sub-cell of both σ and τ,
3) Any compact subset of $\mathcal{K}$ intersects only a finite number of cells.

A subset $\mathcal{M} \subset \mathbb{R}^3$ is said to be an n-dimensional combinatorial manifold with boundary (n=0,1,2,3) if it has an n-cell decomposition in which the neighborhood of each point is homeomorphic either to an n-sphere or to an n-semi-sphere. The 0,1,2 and 3 dimensional manifold will be called, respectively, point, curve, surface and volume. A combinatorial stratification of a set $\mathcal{K} \subset \mathrm{R}^3$ is a chosen finite collection of combinatorial submanifolds with boundary such that their union is $\mathcal{K}$ and the intersection of two of its elements

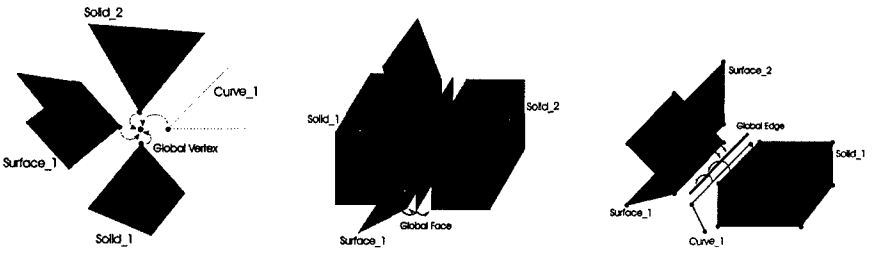


Fig. 1. Global Elements.

belongs to the cellular decomposition. Each manifold in this combinatorial stratification is called a stratum. A stratum could be a point, a curve with or without boundary, a surface with or without boundary or a 3-manifold with boundary.

In this paper, an object is defined as a set $\mathcal{O} \subset \mathrm{R}^3$ endowed with a 3-dimensional combinatorial stratification.

Now we describe the data structure behind the HS-Rep for the cell decomposition and the stratification of an object. The data structure nodes are classified in three types: strata, local cells and global cells.

- Strata nodes (Point, Curve, Surface and Volume) describe the manifold components of the object stratification.
- Local cell nodes represent the cells of a stratum. For instance, a curve has two types of local cell elements, the curve-vertex and the curve-edge. A surface has three types of local cell nodes: surface-vertex, surface-edge and surface-face. Also, there are three kinds of local elements for volumes: volume-vertex, volume-edge and volume-face.
- Global cell nodes (Global Vertex, Global Edge and Global Face) are used to represent the cellular decomposition of the object. Also, global cell are used to identify the local cells of different strata. A global cell is said to be singular if it has more than one local cell associated with it. Thus, on this data structure the singularities are explicitly represented.

In Figure 1 some examples of the use of the global vertex, global edge and global face cells are shown. In Figure 2 we have the hierarchy scheme of the data structure associated with the HS-Rep.

The main characteristic of this new data structure is the explicit representation of the object stratification. The stratification allows the representation of singular objects and manifolds of different dimensions in the same environment. Those manifolds are linked together through the global cells. One advantage of using objects as defined in this paper is that it keeps to a minimum the redundant information stored in each cell.

The representations introduced by Weiler [12] and Gursoz [5] also deal with singular objects (non-manifolds), but they don't identify manifold components.

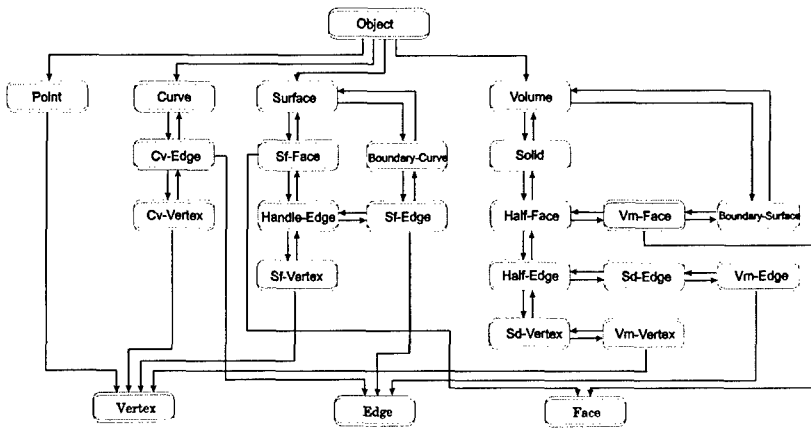


Fig. 2. HS-Rep Data Structure.

A set of operators to build and unbuild an object on this representation, called Morse operators, will now be described. These operators are validated by the Piecewise-Linear Handlebody Theory [10] and they correspond to gluing handles on manifolds with boundary. Morse operators are divided in two groups: local and global operators. Local operators build and unbuild strata. Global operators perform the union of strata.

The local building operators are used to identify two boundary m-cells ($m = 0, 1, 2$) of a respective regular $(m + 1)$-dimensional manifold. The local building operators for curves create an interior vertex by the identification of two boundary vertices, which can be both on the same curve component, or on different connected components.

There are five situations where two boundary edges of surfaces can be identified. For each one a local building operator is defined. These five cases are distinguished by the following criteria: 1) the two boundary edges don't have vertices in common but they are on different surface components; 2) the two boundary edges don't have vertices in common but they are on different boundary curve components of a surface (on this situation, a genus is created on a surface); 3) the two boundary edges don't have vertices in common but they are on the same boundary curve; 4) the two boundary edges have only one vertex in common and, finally, 5) the two boundary edges have two vertices in common. More details on those operators on surfaces can be found in [2].

For 3-manifolds, there are also five situations where two boundary faces can be identified. Each case defines a local building operator for 3-manifolds. These cases are distinguished according to the following criteria: 1) the two boundary faces are on the same connected component of the manifold; 2) the two boundary faces are on the same boundary surface component; the two boundary faces have or have not edges in common. A detailed discussion of these operators for 3-Manifolds can be found in [6].

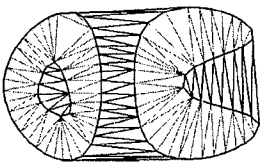


Fig. 3. Edges of $\mathcal{T}$ on a planar section.

The global operators identify m-dimensional global cells ($m = 0, 1, 2$). Those operators make the union of different strata through the use of the global elements on the data structure.

§3. Volumetric Reconstruction from Planar Sections

In this section a heuristic based on the Delaunay Triangulation for the volumetric reconstruction is discussed. This heuristic was introduced in [7] and is now rediscussed in terms of the representation introduced in this paper.

Here the reconstruction process will be restricted to two consecutive planar sections. The object is built from contours in the planar sections by applying the appropriate heuristic to the Delaunay triangulation. These contours are simple polygons that bounds the planar regions to be connected, and can be oriented coherently.

The first phase of the reconstruction process generates a 3-dimensional Delaunay triangulation that contains all edges of the contours on two consecutive slices. This triangulation will be called the restricted Delaunay triangulation of the slices, and will be denoted by $\mathcal{T}$. To obtain such a triangulation, the following algorithm has been devised:

1) Build a 3-dimensional Delaunay triangulation $\mathcal{D}$ using the vertices of all contours,
2) Mark the edges of the contours that are not contained on $\mathcal{D}$,
3) Subdivide all marked edges, inserting new vertices on the contours,
4) Make local modifications on $\mathcal{D}$ to obtain a new Delaunay triangulation that includes those new vertices,
5) Repeat these steps until the triangulation contains all contour edges.

Boissonnat [1] shows that the missing edge subdivision strategy, used in the above algorithm, obtains a Delaunay Triangulation that includes all contour edges.

The second phase of the reconstruction process classifies the edges of $\mathcal{T}$ contained in the planar sections as internal, external or contour edges according whether they are internal, external or on the contours. Figure 3 shows a set of contours and the external, internal and contour edges of $\mathcal{T}$ that are on a planar section.

To generate a model, which satisfy the resampling condition, i.e. whose intersection with the given planes corresponds exactly to the same given contours, it is necessary to identify the connected components and modify the

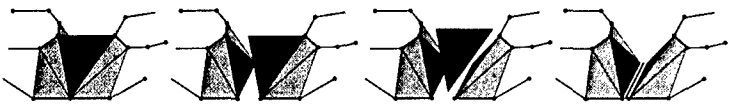


Fig. 4. Singular edge coming from a reverse tetrahedron elimination.



Fig. 5. Tetrahedron subdivision avoiding singular edge creation.

triangulation. For this we will introduce the concept of reverse tetrahedra and geometrically well positioned contours.

A tetrahedron of the triangulation $\mathcal{T}$ is called a reverse tetrahedron if they have edges on different slices which are not contour edges. Two contours on consecutive slices are said to be geometrically well positioned if they are connected by a reverse tetrahedron in $\mathcal{T}$.

Intuitively, it is appropriate to maintain on the same connected component contours based on distinct slices which are geometrically well positioned. In the heuristic introduced in [7], reverse tetrahedra play an essential role on the 3-manifold components definition because they identify when two contours are connected to each other.

A singular edge on $\mathcal{T}$ is defined as an edge whose associated link is not homeomorphic either to a sphere or to a semi-sphere on the corresponding 3-dimensional manifold.

Next we can use the representation introduced above to deal with the branching problem. We propose a heuristic using singular edges which at the end generates a triangulated manifold between the slices:

1) Remove all tetrahedra with at least one external edge. The removal of one tetrahedron may generate a singular edge, see Figure 4.

2) Identify singular edges.

   a. If the singular edge is interior to the contour, reinsert the corresponding reverse tetrahedron, subdivide its external edge and push the new vertex to a position in between the slices, see Figure 5. The role of this translation is to guarantee the resampling condition.

   b. If the singular edge is on a contour, split the connected components as in Figure 6.

Finally, the whole object is reconstructed by putting together the objects built between consecutive slices.

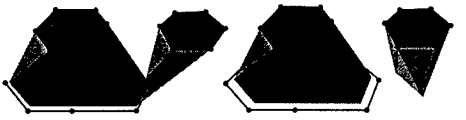


Fig. 6. Avoiding edge singularity on a contour edge.

§4. HS-Rep Applications to Volumetric Reconstruction

The algorithm of Nonato and Tavares [7] has no explicit data structure dealing with edge singularities. Thus, the main contribution of this section is to present several instances where the HS-Rep representation simplifies the reconstruction process.

Initially, all contours are created on the data structure using the building operators for curves. The vertices of these contours are used to build the initial 3-dimensional Delaunay triangulation $\mathcal{D}$. After that, the contour vertices are identified with the vertices of $\mathcal{D}$ through the global vertex operator.

To verify that a contour edge is on the triangulation $\mathcal{D}$, one has to look at the star of each global vertex and check for incidence to verify if this contour edge is on the boundary surface of the volume. If the contour edge is on the boundary surface then it is associated with the corresponding contour edge on the slice curve by using a global-edge building operator. Otherwise, the contour edge must be subdivided and the Delaunay triangulation will be locally modified to include this new point. This process will continue until the triangulation $\mathcal{T}$, which contains all contour edges, is obtained. The non-contour edges whose vertices are on the same slice can be classified either as internal or external traversing the list of edges of the boundary surface of $\mathcal{T}$, which is then explicitly represented on the data structure.

Section 3 points out that the first step in the identification of the connected components is the elimination of the tetrahedra with at least one external edge. To remove a tetrahedron, split its internal faces into boundary faces using the local Morse operators for 3-manifolds. To reconstruct it as a manifold, singular edges have to be detected. Global singular edges are detected by performing a counting on the number of incident 3-manifold strata.

Now local building operators for 3-manifolds are used to insert tetrahedra and subdivide its edges. The new vertices added in this subdivision are translated to a position inbetween the slices. When the singularity is a contour edge, a global operator is used to split the manifold components.

Finally, the objects obtained on consecutive slices are glued together. The process of gluing those objects consists in applying local building operators to all boundary faces on the contour interior.

The Handle-Strata computational environment is suitable for dealing with either the strategy given by Nuages [8] or Nonato and Tavares [7]. Moreover, this representation is well suited for4 integrating different techniques under the same common topological kernel. Issues like graphics interface, visualization, objects physical properties, deformations, and so on, can now be addressed as



Fig. 7. Reconstruction of the bitorus' slices using Nonato and Tavares heuristic.



Fig. 8. Reconstruction of bitorus' slices using Nuages.

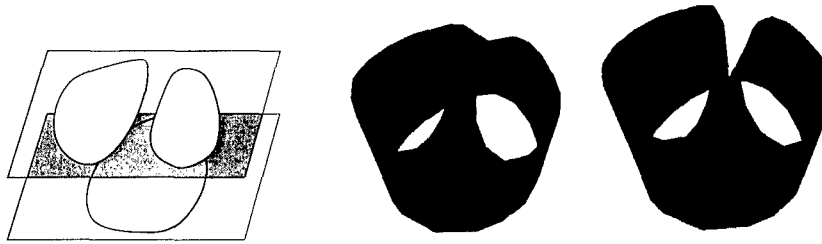


Fig. 9. Reconstruction using, respectively, Handle-Strata and Nuages.

attributes or applications of the Handle-Strata representation.

The examples below come from three slices of a bitorus, in which the bottom and the top slice have two curves and the intermediate slice has only one curve. For these images, only the boundary faces not on the slices are visualized. In Figures 7 and 8 we show the input slices and two views of the models reconstructed using, respectively, the proposed algorithm and that of Nuages.

Nuage's reconstruction inserts edge singularities at an intermediate level. The reconstruction using *Handle-Strata* avoided that singularity through tetrahedra insertion and subdivision.

The second example take two slices of a torus, in which the one on the top has two contours that are geometrically well positioned with the unique contour on the bottom slice. The three pictures of Figure 9 show, the slices, our reconstructed model, and Nuage's result, respectively.

The intersection of the bottom plane with the model created with that of Nuages is not the original curve, i.e, it does not satisfy the resampling condition. The new heuristic creates a saddle in order to avoid this singularity.



Fig. 10. Spine Vertebra Reconstruction.

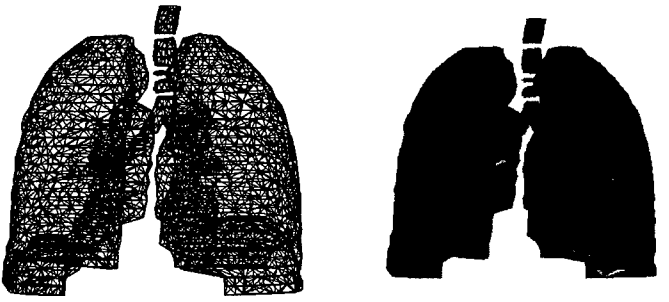


Fig. 11. Lung Reconstruction.

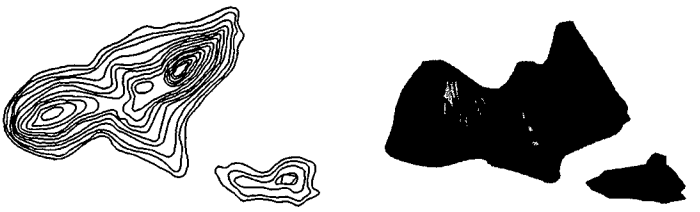


Fig. 12. Sugar Loaf Reconstruction.

The execution time and the number of tetrahedra in the final objects for both algorithms are essentially the same.

§6. Examples

Figure 10 shows the reconstruction of a Spine Vertebra. Figure 11 shows the reconstruction of a lung. Figure 12 shows an example of a terrain reconstruction given by its contour levels.

Acknowledgments. We would like to thank Gill Barequet from Tel Aviv University for providing us with the Spine Data, Bernhard Geiger from INRIA - Sophia Antipolis - France for the Lung Data and Marco Grivet from PUC-Rio for the Sugar Loaf Data. Also we like to thank the unknown referee for the valuable suggestions on how to improve the paper.

References

1. Boissonnat, J.–D., Shape Reconstruction from planar cross sections, Computer Vision, Graphics and Image Processing **44** (1988), 1–29.
2. Castelo, A., H. Lopes and G. Tavares, Handlebody representation for surfaces and Morse operators, in *Curves and Surfaces in Computer Vision and Graphics III*, J. Warren (ed.), SPIE Proceedings Series, Boston, 1992, 270–283.
3. Edelsbrunner, H., E. P. Mücke, Three-dimensional alpha shapes, ACM Transactions on Graphics **13 (1)** (1994), 43–72.
4. Forman, R., Morse theory for cell complexes, Advances in Mathematics **134** (1998), 90–145.
5. Gursoz, L., Y. Choi and F. Prinz, Vertex-based representation of non-manifold boundaries, in *Geometric Modeling for Product Engineering*, M. Wozny, J. Turner and K. Preiss (eds.), IFIP, 1990, 107–130.
6. Lopes, H., and G. Tavares, Structural operators for modeling 3-manifolds, in *Proceedings of the Fourth ACM SIGGRAPH Simposyum on Solid Modeling and Its Applications*, C. Hoffmann, W. Brossvort (eds.), ACM Press, Atlanta, 1997, 9–17.
7. Nonato, L.G. and G. Tavares, Volumetric manifold reconstruction of planar sections, manuscript.
8. Nuages – `http://www-sop.inria.fr/prisme/logiciel/nuages.html`
9. Pesco, S., Geometric modeling of polyhedral singular spaces, Ph.D. Thesis (In portuguese), Department of Mathematics, PUC-Rio, 1997.
10. Rourke, C.P. and B. J. Sanderson, *Introduction to Piecewise-linear Topology*, Springer Verlag, 1982.
11. VolVis – `http://www.cs.sunysb.edu/ vislab/volvis_home.html`
12. Weiler, K., Topological structures for geometric modeling, Ph.D. Thesis, Rensselaer Polytechnic Institute, 1986.

H. Lopes, L. Nonato, S. Pesco and G. Tavares
Pontifícia Universidade Católica do Rio de Janeiro
Departamento de Matemática
Rua Marquês de São Vicente, 225, Gávea
Rio de Janeiro, RJ, Brazil, CEP:22.453-900
{lopes,nonato,pesco,geovan}@mat.puc-rio.br

Linear Envelopes for Uniform B–spline Curves

David Lutterkort and Jörg Peters

Abstract. We derive an efficiently computable, tight bound on the distance between a uniform spline and its B–Spline control polygon in terms of the second differences of the control points. The bound yields a piecewise linear envelope enclosing the spline and its control polygon. For quadratic and cubic splines the envelope has minimal possible width at the break points, and for all degrees the maximal width shrinks by a factor of 4 under uniform refinement. We extend the construction to tight envelopes for parametric curves.

§1. Motivation and Overview

The central feature that allows reasoning about nonlinear piecewise polynomials is the fact that a spline is closely outlined by its B-spline control polygon. The efficiency of many applications depends crucially on a good estimate of the distance separating spline and control polygon. For rendering, a refined control polygon is rendered instead of the curve itself. For curve-intersection an efficient and robust technique is to recursively refine and intersect control polygons [2]. Assessing the exactness of these operations requires a uniform, linear bound on the distance of the curve and its (refined B-spline) control polygon. The efficiency is improved if the effect of the refinement can be predicted rather than just measured. Of the two classical bounding contructs, axis-aligned min-max coefficient boxes and the convex hull, the first yields only a loose envelope and neither yields *a priori* estimates.

This paper introduces quantitative bounds that can be computed more efficiently than convex hulls, and yield a simple piecewise linear envelope enclosing spline and control polygon (see Figure 1) whose maximal width contracts to 1/4th when the knot spacing is halved. The computation of the envelope of a degree d spline consists of computing the second differences of its control points and looking up or calculating $d-1$ constants, the values of a fixed set of splines. The sum of the constants, $(d+1)/24$, provides a

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 239–246.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

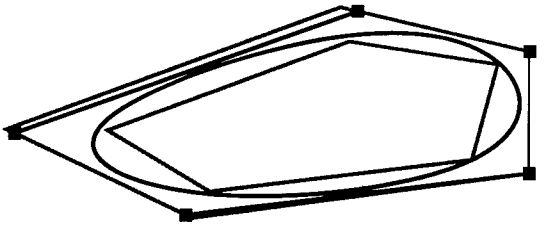


Fig. 1. A cubic curve *(black)* and its control points *(black squares)*. The envelope *(grey)* is constructed with the bound from Theorem 2.

second, even simpler, but generally much coarser bound (Figure 2). Both bounds are piecewise linear with breaks at the corners of the control polygon and are *sharp for quadratic and cubic splines* in the sense that at every corner of the control polygon the distance between the spline and polygon is matched exactly.

This paper derives these bounds for *functions* and establishes the convergence of the bound under uniform refinement; the bounds are then applied to curves to obtain *localized* envelopes.

§2. Notation

A scalar-valued piecewise polynomial p of degree d is a uniform B–spline if it can be represented as

$$p = \sum_{k \in \mathbb{Z}} b^k N^k, \quad b^k \in \mathbb{R}, \quad N^k = N(\cdot - k),$$

where N is the B-spline of degree d supported on the interval $[0, d+1)$ and with the uniform knot sequence $\mathbb{Z}$ (c.f. [2]). For simplicity, we assume that both the control point sequence and the knot sequence are biinfinite.

The control polygon ℓ of p is the piecewise linear interpolant of the control points b^k at the Greville abscissae

$$t_k^* = k + (d+1)/2.$$

Over the interval $[t_k^*, t_{k+1}^*]$, the control polygon is $\ell(t) = \mathrm{L}_k(t\,; b^k, b^{k+1})$ where we denote the line segment from (t_k^*, a_1) to (t_{k+1}^*, a_2) by

$$\mathrm{L}_k(t\,; a_1, a_2) = a_1(t_{k+1}^* - t) + a_2(t - t_k^*).$$

The linear interpolant of a function f over this interval will be abbreviated as $\mathrm{L}_k(f) = \mathrm{L}_k(\cdot\,; f(t_k^*), f(t_{k+1}^*))$. The (centered) second differences of b are defined as

$$\Delta_2 b^i = b^{i-1} - 2b^i + b^{i+1}.$$

The first and the last basis function that are supported on $[t_k^*, t_{k+1}^*]$ are $N^{\underline{k}}$ and $N^{\overline{k}}$ with

$$\underline{k} = k + 1 - \lfloor d/2 \rfloor, \qquad \overline{k} = k - 1 + \lfloor d/2 \rfloor.$$

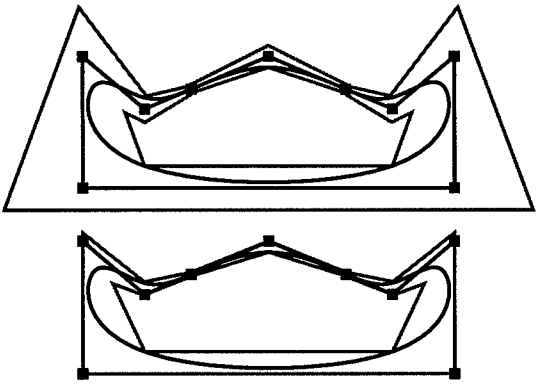


Fig. 2. A cubic curve *(black)* and its control points *(black squares)*. On top the envelope *(grey)* is constructed with the bound from Theorem 3, on the bottom from the tighter bounds of Theorem 2.

§3. Uniform B–splines Bounds

The key observation for deriving the bounds is that the difference between a uniform B–spline p and its control polygon can be factored into two parts: the second differences of the control polygon and splines β_{ki}, which are independent of p.

Theorem 1. *Over the interval $[t_k^*, t_{k+1}^*]$, the difference between a uniform B–spline p and its control polygon ℓ is given by*

$$p - \ell = \sum \Delta_2 b^i \, \beta_{ki}, \qquad \beta_{ki} = \begin{cases} \sum_{j=-\infty}^{i} (i-j) N^j & i \le k \\ \sum_{j=i}^{\infty} (j-i) N^j & i > k. \end{cases}$$

The functions β_{ki} are non-negative and convex on the interval $[t_k^*, t_{k+1}^*]$ and $\beta_{ki}(t_k^*) > 0$ if and only if $i \in [\underline{k}, \overline{k}]$.

Proof: We write $p - \ell$ over $[t_k^*, t_{k+1}^*]$ as

$$\sum_{i=\underline{k}}^{\overline{k}} b^i \, \alpha_{ki} = \sum_{i=\underline{k}}^{\overline{k}} b^i \left(N^i(t) - \mathrm{L}_k(t; \delta_{ik}, \delta_{i,k+1}) \right),$$

where $\delta_{ik} = 1$ if $i = k$ and 0 otherwise. We show that $\alpha_{ki} = \Delta_2 \beta_{ki}$: the partition of unity $\sum_i N^i = 1$ implies that $\sum_i \alpha_{ki} = 0$ and the linear precision of B–Splines, $\sum_i t_i^* N^i(t) = t$ implies on the interval $[t_k^*, t_{k+1}^*]$ that $\sum_i i \alpha_{ki} = 0$. Hence, for any i, $\sum_j (j-i) \alpha_{kj} = 0$.

For $i > k$,

$$\beta_{ki} = \sum_{j=i}^{\overline{k}} (j-i) N^j - \sum_{j=\underline{k}}^{\overline{k}} (j-i) \alpha_{kj} = \sum_{j=\underline{k}}^{i} (i-j) \alpha_{kj}$$

so that $\beta_{ki} = \sum_{j=\underline{k}}^{i}(i-j)\alpha_{kj}$ for any i. It is now straightforward to verify that $\Delta_2\beta_{ki} = \alpha_{ki}$ and summation by parts yields

$$p - \ell_k = \sum_{i=\underline{k}}^{\overline{k}} b^i\,\alpha_{ki} = \sum_{i=\underline{k}}^{\overline{k}} b^i\,\Delta_2\beta_{ki} = \sum_{i=\underline{k}}^{\overline{k}} \Delta_2 b^i\,\beta_{ki}.$$

The functions β_{ki} are non-negative since their B–spline coefficients are non-negative. The convexity of the β_{ki} over $[t_k^*, t_{k+1}^*]$ follows from the convexity of their B–spline control polygons: for $i > k$, the part of the control polygon of β_{ki} that influences β_{ki} over $[t_k^*, t_{k+1}^*]$ lies on the function $\max\{\cdot - t_i^*, 0\}$ while for $i \le k$ it lies on $\max\{t_i^* - \cdot, 0\}$. In both cases, the control polygon of β_{ki}, and hence β_{ki}, is non-negative and convex. □

Theorem 1 immediately gives us a piecewise linear envelope on $p - \ell$:

Theorem 2. *Over the interval $[t_k^*, t_{k+1}^*]$, the difference between a uniform B–spline p and its control polygon ℓ is bounded by*

$$\mathrm{L}_k\left(\sum \Delta_2^- b^i\,\beta_{ki}\right) \le p - \ell \le \mathrm{L}_k\left(\sum \Delta_2^+ b^i\,\beta_{ki}\right),$$

where $\Delta_2^+ b^i = \max\{\Delta_2 b^i, 0\}$ and $\Delta_2^- b^i = \min\{\Delta_2 b^i, 0\}$.

Proof: We have from Theorem 1

$$p - \ell = \sum_i \Delta_2 b^i\,\beta_{ki} = \sum_i \Delta_2^+ b^i\,\beta_{ki} + \sum_i \Delta_2^- b^i\,\beta_{ki}.$$

The positivity of the β_{ki} implies that the first sum on the right–hand side is positive and the second is negative and therefore

$$\sum_i \Delta_2^- b^i\,\beta_{ki} \le p - \ell \le \sum_i \Delta_2^+ b^i\,\beta_{ki}.$$

Since the β_{ki} are convex over $[t_k^*, t_{k+1}^*]$, they can be bounded linearly to yield the bound of Theorem 2. □

An even simpler envelope can be derived by bounding the sum of the β_{ki} at t_k^* by the constant $(d+1)/24$.

Theorem 3. *Over the interval $[t_k^*, t_{k+1}^*]$, the difference between a uniform B–spline p and its control polygon ℓ is bounded by*

$$|p - \ell| \le \frac{d+1}{24}\,\mathrm{L}_k(\,\cdot\,; \|\Delta_2 b\|_k, \|\Delta_2 b\|_{k+1}),$$

where $\|\Delta_2 b\|_k = \max\{|\Delta_2 b^i| : i \in [\underline{k}, \overline{k}]\}$. If $d = 2$ or $d = 3$ the relation holds with equality at the t_k^*.

Proof: By Theorem 1 and the convexity of the β_{ki} over $[t_k^*, t_{k+1}^*]$, we have

$$\begin{aligned} |p - \ell| &\le \sum_i |\Delta_2 b^i|\,\beta_{ki} \le \mathrm{L}_k\left(\sum_i |\Delta_2 b^i|\,\beta_{ki}\right) \\ &\le \mathrm{L}_k(\,\cdot\,; \|\Delta_2 b\|_k \sum_i \beta_{ki}(t_k^*), \|\Delta_2 b\|_{k+1} \sum_i \beta_{ki}(t_{k+1}^*)). \end{aligned} \tag{1}$$

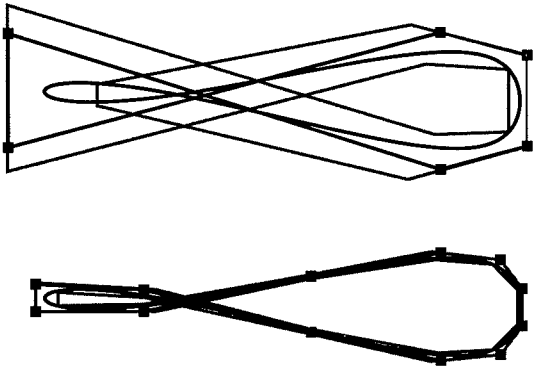


Fig. 3. A self–intersecting quartic curve *(black)* and its control points *(black squares)*. The envelopes *(grey)* are constructed with the bound from Theorem 2. The envelope converges rapidly to the curve as the comparison of the original envelope, top, and the envelope after one step of uniform refinement, bottom, shows.

The theorem follows if we can show that $\sum_i \beta_{ki} \leq (d+1)/24$:

$$\sum_i \beta_{ki} = \sum_{i=\underline{k}}^{\overline{k}} \sum_{j=\underline{k}}^{i} (i-j)\,\alpha_{kj} = \sum_{j=\underline{k}}^{\overline{k}} \sum_{i=j}^{\overline{k}} (i-j)\,\alpha_{kj} = \sum_j \alpha_{kj} \sum_{i=0}^{\overline{k}-j} i$$

$$= \sum_j \binom{j-\overline{k}}{2} \alpha_{kj} = \sum_j \binom{j}{2} \alpha_{kj} = \sum_j \binom{j-k}{2} N^j =: z.$$

Regardless of the degree of p, z is the quadratic polynomial

$$z(t) = \frac{t^2}{2} - \frac{t_k^* + t_{k+1}^*}{2} t + \frac{1}{2} \left(\frac{t_k^* + t_{k+1}^*}{2} \right)^2 + \frac{d-2}{24}.$$

Since z is a positive and convex function, z attains its maximum over $[t_k^*, t_{k+1}^*]$ at one of the endpoints of the interval. Its values there are

$$z(t_k^*) = z(t_{k+1}^*) = \frac{d+1}{24}$$

and hence $z(t) = \sum_i \beta_{ki} \leq (d+1)/24$ for all $t \in [t_k^*, t_{k+1}^*]$.

The number of β_{ki} that are nonzero at t_k^* is $d-1$ for d even and $d-2$ for d odd, i.e. only β_{kk} is nonzero at t_k^* if p is quadratic or cubic. But then all inequalities of equation (1) become equalities as claimed. □

Computing the bounds

To compute the bounds for quadratics or cubics no B–spline evaluation is required, since only $\beta_{kk}(t_k^*) = (d+1)/24$ is nonzero. For $d > 3$, it suffices to look up tabulated values $\beta_{ki}(t_k^*)$ for $-\lfloor d/2 \rfloor < i < \lfloor d/2 \rfloor$. Forming the inner products of Theorem 2 and Theorem 3 at t_k^* is straightforward.

§4. Uniform Refinement

An important operation on B–splines is the refinement of the knot sequence or knot insertion. Knot insertion changes the representation of the piecewise polynomial p over the original knot sequence to one over a larger knot sequence and reduces the distance between spline and control polygon (c.f. Figure 3).

After halving the distance between knots the new control points $\hat{b}^k$ in $p(t) = \sum_k b^k N^k(t) = \sum_k \hat{b}^k N^k(2t)$ are given by

$$\hat{b}^{2i} = 2^{-d} \sum_{j=0}^{\lceil d/2 \rceil} \binom{d+1}{2j} b^{i-j}, \qquad \hat{b}^{2i+1} = 2^{-d} \sum_{j=0}^{\lceil d/2 \rceil} \binom{d+1}{2j+1} b^{i-j}. \tag{2}$$

Theorem 4. *The second differences $\Delta_2 \hat{b}^i$ of the refined control polygon are bounded by the second differences $\Delta_2 b^i$ of the original control polygon*

$$\max_i |\Delta_2 \hat{b}^i| = \frac{1}{4} \max_i |\Delta_2 b^i|.$$

Proof: The second derivative p'' of p is given by

$$p''(t) = \sum_k \Delta_2 b^{k-1} N^k_{d-2}(t) = \sum_k \Delta_2 \hat{b}^{k-1} N^k_{d-2} k(2t),$$

which means that the $\Delta_2 \hat{b}^i$ can be obtained from the $\Delta_2 b^i$ via (2) as

$$2^d \Delta_2 \hat{b}^{2i} = \sum_j \binom{d-1}{2j-1} \Delta_2 b^{i-j}, \quad 2^d \Delta_2 \hat{b}^{2i+1} = \sum_j \binom{d-1}{2j} \Delta_2 b^{i-j}.$$

The proof follows from $\sum_j \binom{d-1}{j} = 2^{d-1}$ and $\sum_j \binom{d-1}{2j} = \sum_j \binom{d-1}{2j-1} = 2^{d-2}$. □

Theorem 4 yields the following *a priori* estimate on the number of subdivisions σ needed to bring spline and control polygon within a given distance ε:

$$\sigma(p, \varepsilon) = \lceil \log_4 \frac{(d+1)\|\Delta_2 b\|}{24\varepsilon} \rceil.$$

Examples: For quadratic B–Splines, uniform refinement is called Chaikin's algorithm, and

$$\hat{b}^{2i} = 2^{-2}(3b^{i-1} + b^i), \qquad \hat{b}^{2i+1} = 2^{-2}(b^{i-1} + 3b^i).$$

This yields

$$\Delta_2 \hat{b}^{2i} = \Delta_2 \hat{b}^{2i-1} = \frac{1}{4} \Delta_2 b^{i-1},$$

i.e. every second difference is guaranteed to decrease by a factor of four. Similarly, for cubic B–Splines we have

$$\hat{b}^{2i} = 2^{-3}(b^{i-2} + 6b^{i-1} + b^i), \qquad \hat{b}^{2i+1} = 2^{-3}(4b^{i-1} + 4b^i),$$

and

$$\Delta_2 \hat{b}^{2i} = \frac{1}{4} \Delta_2 b^{i-1}, \qquad \Delta_2 \hat{b}^{2i+1} = \frac{1}{4} \frac{\Delta_2 b^{i-1} + \Delta_2 b^i}{2}.$$

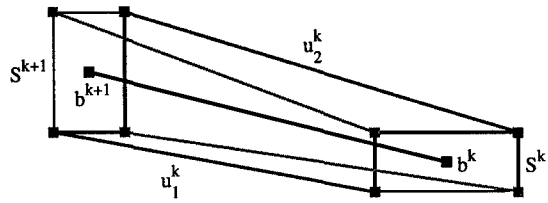


Fig. 4. Constructing the envelope of a curve from the bounding rectangles S^k and S^{k+1}: only the outer line segments $\boldsymbol{u}_1^k$ and $\boldsymbol{u}_2^k$ are part of the convex hull of $S^k \cup S^{k+1}$ and the envelope.

§5. Curve Envelopes

A parametric curve $\boldsymbol{p}$ is in uniform B–spline form if $\boldsymbol{p} = \sum_j \boldsymbol{b}^j N^j$ where the $\boldsymbol{b}^j \in \mathbb{R}^n$ are the control points of $\boldsymbol{p}$ and the uniform B–spline basis N^j is defined as in Section 2. The curve $\boldsymbol{p}$ is closed if the control point sequence $(\boldsymbol{b}^j)$ is periodic.

The functional bounds are applied componentwise to parametric curves. Then each control point and the curve point corresponding to its Greville abscissa lie in a box whose width in the ith component is the bound in the ith component. It is now convenient to restate the bounds from Theorems 2 and 3 more abstractly as

$$\underline{e}(t) \le p(t) - \ell(t) \le \overline{e}(t) \qquad \text{for } t \in [t_k^*, t_{k+1}^*]. \tag{3}$$

For curves $\boldsymbol{p}$, the bound in the i–th component is denoted by $\underline{e}_i \le \boldsymbol{p}_i - \ell_i \le \overline{e}_i$.

By (3), $\boldsymbol{p}(t_k^*)$ is located in the axis–aligned box S^k,

$$S^k = \{\boldsymbol{x} \mid \underline{e}_i(t_k^*) \le \boldsymbol{x}_i - \boldsymbol{b}_i^k \le \overline{e}_i(t_k^*) \quad \text{for all } i = 1, \ldots, n\}.$$

Each point of the curve segment $\boldsymbol{p}(t)$, $t \in [t_k^*, t_{k+1}^*]$, lies in a box $S(t)$, that by the linearity of $\underline{e}$ and $\overline{e}$ is a convex combination of S^k and S^{k+1}:

$$S(t) = \mathrm{L}_k(t; S^k, S^{k+1}).$$

The curve segment is therefore contained in the union of all $S(t)$, $t \in [t_k^*, t_{k+1}^*]$, which is the convex hull H^k of the corners of S^k and S^{k+1}. To be specific, we discuss the case of planar curves.

Enveloping planar curves

Let $\boldsymbol{v}_i^k$, $i = 1, \ldots, 4$, be the line segments connecting corresponding corners of S^k and S^{k+1}; that means $\boldsymbol{v}_1^k$ connects the lower left corner of S^k to the lower left corner of S^{k+1}, $\boldsymbol{v}_2^k$ connects the lower right corner of S^k to the lower right corner of S^{k+1} etc. as in Figure 4.

H^k consists of parts of the boundaries of S^k and S^{k+1} and exactly two additional line segments $\boldsymbol{u}_1^k$ and $\boldsymbol{u}_2^k$ chosen from the $\boldsymbol{v}_i^k$. Since $\boldsymbol{u}_1^k$ and $\boldsymbol{u}_2^k$ are

part of the convex hull H^k, they do not intersect the interiors of S^k and S^{k+1}. We do not need to actually compute intersections of the $\boldsymbol{v}_i^k$ and S^k, S^{k+1} to select $\boldsymbol{u}_1^k$ and $\boldsymbol{u}_2^k$: since S^k and S^{k+1} are axis–aligned it suffices to look at the signs of the slopes of the $\boldsymbol{v}_i^k$. The $\boldsymbol{u}_i^k$ are separated by the line from $\boldsymbol{b}^k$ to $\boldsymbol{b}^{k+1}$; we call the one lying to the left of this line $\boldsymbol{u}_1^k$ and the one lying to the right of this line $\boldsymbol{u}_2^k$.

The sets $U_i = \{\boldsymbol{u}_i^k\}$ are not yet polylines: consecutive line segments $\boldsymbol{u}_i^k$ and $\boldsymbol{u}_i^{k+1}$ may intersect or not touch at all. But note that the line extending $\boldsymbol{u}_i^k$ always intersects the one extending $\boldsymbol{u}_i^{k-1}$. We obtain a proper polyline W_i with exactly one line segment for each control point of $\boldsymbol{p}$ by taking this intersection as starting point and the intersection with the line through $\boldsymbol{u}_i^{k+1}$ as the end point of W_i. The polylines W_1 and W_2 then form a *local* envelope of $\boldsymbol{p}$: the curve–piece $\boldsymbol{p}([t_k^*, t_{k+1}^*])$ lies in the quadrangle spanned by the k–th pieces of W_1 and W_2.

References

1. de Boor, C., B(asic)-spline basics; in *Fundamental Developments of Computer Aided Geometric Modeling*, Les Piegl (ed.), Academic Press, London, 1993, 27–49.
2. Hoschek, J., and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, Translated from the 1992 German edition by Larry L. Schumaker, A K Peters Ltd., Wellesley, MA, 1993.

David Lutterkort
Dept Computer Sciences
Purdue University
West Lafayette, IN, 47905-1398 USA
lutter@cise.ufl.edu

Jörg Peters
Dept C.I.S.E., CSE Bldg
University of Florida
Gainesville, FL 32611-6120, USA
jorg@cise.ufl.edu

n-sided Surfaces: a Survey

Pierre Malraison

Abstract. The paper surveys techniques for filling in n–sided regions, where $n > 4$. The two major classes of methods examined are: 1) to fill in the hole with 4 and/or 3 sided patches, 2) to create a single surface. The multi–patch approaches differ in terms of the degree of the patches and the cross–patch continuity. The single surface approaches are either rational surfaces (which can be expressed in terms of base points) or non-rational, both cases having a number of variants.

§1. Introduction

The problem being considered is:

> *Given n curves $C_1, \ldots, C_n$ whose endpoints match, i.e (if we say $C_0 = C_n$) the end of C_{i-1} is the start of C_i, fill in the hole bounded by the C_i, possibly satisfying some additional boundary conditions.*

For example, in blending, the C_i are the edges of faces, and the filling surface or surfaces must be smooth across the edges.

I will be looking at the case $n > 4$. The problem with no boundary conditions arises in the *cover* command in the ACIS [1] software libraries. ACIS also supports vertex blends using Charrot [6].

This paper extends the survey Malraison [38]. Other surveys include: Nasri [40], Cavaretta [4], Sederberg [57] and Dyn [9] for general overviews of subdivision, Dyn [10] for a review of John Gregory's contributions to the field, Varady [65] is a review of n-sided patches, Gregory [13] and Gregory [16] are surveys on n-sided patches by Gregory and others, Varady [60] specific to vertex blends and Vida [66] discusses blends in general with a section on n-sided issues.

§2. Subdivision

Subdivision is a much broader topic than I can cover here. The basic process is to generate a surface by starting with a polygonal approximation P_0 and having a process which from P_i creates P_{i+1}. The n-sided problem arises

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 247–255.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

in this context when the individual polygons formed by the subdivision process are not 4-sided. Hermann [21] and Ball [2] use subdivision to explicitly fill in n-sided holes, while Wang [69,68] uses a single patch method of Varady [65,64] to fill in an n-sided hole arising in the course of subdivision. Levin [31,32] applies a *combined* subdivision scheme to solve the n-sided problem with cross-tangency constraints. Nasri [41–44] uses subdivision to provide both a source of and solution for n-sided problems.

§3. Multiple Patches

One solution to the problem is to take the n-sided region and subdivide it once into triangular or 4-sided regions, and then fill those with standard surface types. The main difficulty with this approach is ensuring the internal smoothness of the resulting network of patches. Peters [45] discusses the problems for doing a C^K join, Varardy [62] looks at curvature matching, and Hall [20,19] looks at the situation when the pieces are Gregory patches. Some of the other approaches are summarized in Table 1.

Boundary	Degree	Continuity	Reference
Mesh points	Bicubic	C^1	[47]
Planes	Biquintic	C^1	[72]
Polygon	Cubic	C^1	[8,3]
Quartic	Bicubic	C^1	[48–50]
Cubic	Quartic	G^1	[52]
Cubic	Quartic triangular	C^1	[36,37]
Quintic	Quintic	GC^2	[74]
Quintic	$2k^2+3k+1$	GC^k	[76]
Quintic	Biquintic	GC^2	[17]

Tab. 1. Multiple patches.

Bangert [3], and Peters [48,49,50] are triangle-based spline methods. Peters [51] adds a hierarchical structure which supports interactive modeling. Some other multi-patch approaches do not fit into the above table. Sone [58,59] subdivides an n-sided hole into quadrilaterals and uses Gregory patches. Hsu [24] uses a blend between two edges to fill in a triangular subset of the n-sided hole, then continues to subdivide the remaining pieces so that the entire hole is filled in with multiple blend surfaces. This approach is different from the usual multiple patch approach as it may require multiple subdivisions to arrive at regions suitable for blending. Varady [65,64] uses multiple patches for setback blends.

§4. Single Patches

Filling in an n-sided hole with a single patch is an easier approach in a solid modeling environment since only one face need be constructed. The issue

here is unusual (*i.e.*, non-rectangular) parametric domains and internal shape control. Hall [20] discusses the control of Gregory patches [13].

For the case where the surface is rational (*i.e.*, $f(u,v) = \frac{p(u,v)}{q(u,v)}$), Warren [71] shows that several different methods are all variants of rational surfaces with base points ($p(u,v) = q(u,v) = 0$). Since base points are singularities, they may occur either on the boundary of the domain or outside. S-patches go from a polygon through an n-simplex : $P \underrightarrow{L} \Sigma \underrightarrow{B} R^3$. Those variants are summarized in Table 2.

Basepoints	Variant	Reference
Boundary		
	$2n$	[64]
	up to 8 sides	[70]
	manifold charts	[12]
	5,6 sides	[54]
	includes holes	[29,30]
External S-Patches:	domain is n-simplex	
	original	[35,34]
	modifies B	[33]
	modifies L	[25,26]
Gregory-like:	Polygonal domain	
	pentagon	[5]
	arbitrary n	[6]

Tab. 2. Single patch.

Gregory [15] starts with a larger problem: interpolating an arbitrary mesh. The solution is to interpolate the "edges" by rational splines to create polygonal curved regions, then extend the splines into strips, and blend the strips into the interior using the same technique as Charrot [5].

The other principal method is to generate a rational surface using a Bezier-like approach by constructing non-rectangular control nets. The boundaries are considered as the edges of Bezier surface patches so higher cross boundary smoothness can be obtained by having the internal control net reflect the adjacent surface control net.

Sabin [53] uses quadratic functions to fill in three-sided and five-sided patches. In Sabin [56] the same technique is applied to a 2-sided patch. Hosaka [23] does the same thing using quadratics and cubics for three-sided, five-sided and six-sided patches. Their solution is described in the general n-sided setting. Zheng [73] extends those two approaches by using higher degrees for higher numbers of sides.

Karčiauskas [27,28] and Zube [77,78] provide a unifying approach similar to Warren [71] for these rational cases by looking at toric varieties.

§5. Other n-sided Things

Two papers address a problem which arises in a more global setting from a classical result in topology.

Theorem. [39]. *If M is a compact connected 2-manifold, M is a 2-sphere with h handles and m cross caps.*

This theorem implies M can be represented by a polygon with edges identified: e.g. a torus is $ABA^{-1}B^{-1}$. Ferguson [11] applies this result to model objects with a single surface. Wallner [67] applies the same idea to *orbifolds*: surfaces defined as images of group actions. These techniques fall into the scope of this paper insofar as the marked polygon defining the surface is an n-sided object in parameter space.

§6. Conclusions

For single surface patches, Warren [71], Karčiauskas [28], and Zube [78] show that the approaches used so far are variations on two main themes. For multi-patch and subdivision methods no such unifying concept has been presented, although the basic notion of subdivision is arguably the unfiying theme of that approach.

Acknowledgments. I would like to thank Malcolm Sabin for inviting me to participate in the minisymposium, and Jorg Peters for suggesting the tabular approach which helped keep this paper within the page limits. The opinions expressed in this paper are the author's and do not represent the views of Spatial Technology, Inc. A web page for updating the bibliography is available at http://www.geocities.com/Tokyo/7970/nside.htm

References

I have marked references not in [38] with an asterisk.

1. ACIS$^{(r)}$ Test Harness User's Guide.
*2. Ball, A. A. and D. J. T. Storry, Design of an n-sided surface patch from Hermite boundary data, Comput. Aided Geom. Design **6** (1989), 111–120.
*3. Bangert, C. and H. Prautzsch, Quadric splines, Comput. Aided Geom. Design **16** (1999), 497–515.
4. Cavaretta, A. S. and C. A. Michelli, The design of curves and surfaces by subdivison algorithms, in *Mathematical Methods in CAGD*, Academic Press, 1989, 115–193.
5. Charrot, P. and J. A. Gregory, A pentagonal surface patch for computer aided geometric design, Comput. Aided Geom. Design **1** (1984), 87–94.
6. Plowman, D. and P. Charrot, A practical implementation of vertex blend surfaces using an n-sided patch, in *Mathematics of Surfaces VI*, Clarendon, 1996, 67–78.

7. Daehlen, M. and V. Skytt, Modelling non-rectangular surfaces using box splines, in *Mathematics of Surfaces III*, Clarendon, 1989, 287–300.

8. Dahmen, W., Smooth piecewise quadric surfaces, in *Mathematical Methods in CAGD*, Academic Press, 1989, 181–194.

9. Dyn, N. J. A. Gregory and D. Levin, Piecewise uniform subdivision schemes, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1995, 111–119.

10. Dyn, N., The work of John Gregory: rational spline interpolation, subdivision algorithms and C^2 polygonal patches, in *Mathematics of Surfaces VI*, Clarendon Press, 1996, 21–42.

11. Ferguson, H. and A. Rockwood, Multiperiperidic functions for surface design, Comput. Aided Geom. Design **10** (1993), 315–328.

*12. Gérot, C., D. Attali and A. Montanvert, From local approximation to a G^1 global representation, in *Curve and Surface Design: Saint-Malo 1999*, Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000, 109–118.

13. Gregory, J. A., n-sided surface patches, in *Mathematics of Surfaces*, Clarendon, 1986, 217–232.

14. Gregory, J. A. and J. M. Hahn, A C^2 polygonal surface patch, Comput. Aided Geom. Design **6** (1989), 69–75.

15. Gregory, J. A. and P. K. Yuen, An arbitrary mesh network scheme using rational splines, in *Mathematical Methods in Computer Aided Geometric Design II*, T. Lyche and L. L. Schumaker (eds.), Academic Press, New York, 1992, 321–329.

16. Gregory, J. A., V. K. H. Lau and J. Zhou, Smooth parametric surfaces and n-sided patches, in *Computation of Curves and Surfaces*, Kluwer, 1990, 457–498.

*17. Gregory, J. A., J. Zhou, Irregular C^2 surface construction using bi-polynomial rectangular patches, Comput. Aided Geom. Design **16** (1999), 423–435.

*18. Hall, R. and G. Mullineux, The Zheng-Ball construction without twist constraints, Comput. Aided Geom. Design **16** (1999), 165–175.

*19. Hall, R. and G. Mullineux, Continuity between Gregory-like patches, Comput. Aided Geom. Design **16** (1999), 197-216.

20. Hall, R. and G. Mullineux, Shape modification of Gregory patches, in *Mathematics of Surfaces VII*, Information Geometers, 1997, 383–408.

21. Hermann, T. and G. Renner, Subdivision of n-sided regions into four–sided patches, in *Mathematics of Surfaces III*, Clarendon Press, 1989, 347–357.

22. Hermann, T., G^2 interpolation of free-form curve networks by biquintic Gregory patches, Comput. Aided Geom. Design **13** (1995), 873–893.

23. Hosaka, M. and F. Kimura, Non-four-sided patch expressions with control points, Comput. Aided Geom. Design **1** (1984), 75–86.

24. Hsu, K. L. and D. M. Tsay, Corner blending of free form n-sided holes, IEEE Computer Graphics and Applications, **18** (1998), 72–78.

25. Karčiauskas, K., On n-sided rational surface patches, preprint.

*26. Karčiauskas, K., Rational m-sided surface patches, in *Mathematics of Surfaces VIII*, Information Geometers, Winchester, 1998, 355–368.

*27. Karčiauskas, K., Algebraic version of convex combination patches, in *Proceedings of a Conference of the Lithuanian Mathematical Society*, 1999.

*28. Karčiauskas, K. and R. Krasauskas, Comparison of different multi-sided surface patches using algebraic geometry, in *Curve and Surface Design: Saint-Malo 1999*, Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000, 163–172.

29. Kato, K., Generation of n-sided surface patches with holes, Computer Aided Design **23** (1991), 676–683.

*30. Kato, K., n-sided surface generation from arbitrary boundary edges, in *Curve and Surface Design: Saint-Malo 1999*, Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000, 173–182.

*31. Levin, A., Combined subdivision schemes for the design of surfaces satisfying boundary conditions, Comput. Aided Geom. Design **16** (1999), 345–354.

*32. Levin, A., Filling N-sided holes by combined subdivision schemes, in *Curve and Surface Design: Saint-Malo 1999*, Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000, 221–228.

*33. Lodha, S. K., Filling n-sided holes, in *Modeling in Computer Graphics: Methods and Applications*, Springer Verlag, 1993, 319–346.

34. Loop, C. and T. D. DeRose, Generalized B–spline surfaces of arbitrary topological type, *SIGGRAPH*, 1990, 346-356.

35. Loop, C., Generalized B–spline surfaces of arbitrary topological type, Ph.D. Thesis, University of Washington, 1992.

36. Loop, C., Smooth spline surfaces over irregular meshes, *SIGGRAPH Proceedings*, 1994, 303–310.

37. Loop, C., A G^1 triangular spline surface of arbitrary topological type, Comput. Aided Geom. Design **11** (1994), 303-330.

*38. Malraison, P., A bibliography for n-sided surfaces, in *Mathematics of Surfaces VIII*, Information Geometers, Winchester, 1998, 419–430.

39. Moise, E. E., *Geometric Topology in Dimensions 2 and 3*, Springer Verlag, 1997, 161 Theorem 4.

40. Nasri, A., Polyhedral subdivision methods for free-form surfaces, ACM Transactions on Graphics **1** (1987), 29–73.

41. Nasri, A., Boundary corner control of recursive subdivision surfaces, Computer Aided Design **23** (1991), 405–411.

42. Nasri, A., Interpolation on irregular networks with normal conditions, Comput. Aided Geom. Design **8** (1991), 89–96.

43. Nasri, A., Curve interpolation in recursively generated B-spline surfaces over arbitrary topology, Comput. Aided Geom. Design **14** (1997), 13–30.

44. Nasri, A., Interpolation of open curves by recursive subdivision surfaces, in *The Mathematics of Surfaces VII*, Information Geometers 1997, 173–188.

45. Peters, J. A., Joining smooth patches at a vertex to form a C^k surface, Comput. Aided Geom. Design **9** (1992), 387–411.

46. Peters, J. A., Smooth free-form surfaces over irregular meshes generalizing quadratic splines, Comput. Aided Geom. Design **10** (1993), 347–361.

47. Peters, J. A., Constructing C^1 surfaces of arbitrary topology using biquadratic and bicubic splines, in *Desiging Fair Curves and Surfaces*, SIAM, 1994, 277–294.

48. Peters, J. A., Biquartic C^1 spline surfaces over irregular meshes, Compuiter Aided Design, **27** (1995), 895–903.

49. Peters, J. A., Spline surfaces from irregular control meshes, Zeitschrift für Angewandte Mathematik und Mechanik, **11** (1996), 69–72.

50. Peters, J. A., Curvature continuous spline surfaces over irregular meshes, Comput. Aided Geom. Design **13** (1996), 101–131.

*51. Peters, J. A., Mesh and surface hierarchies, manuscript.

52. Renner, G., Polynomial n-sided patches, in *Curves and Surfaces*, P.-J. Laurent, A. Le Méhauté, and L. L. Schumaker (eds.), Academic Press, New York, 1991, 407–410.

53. Sabin, M., Non-rectangular surface patches suitable for inclusion in a B–spline surface, in Proceedings Eurographics83, North Holland, 1983, 57–70.

54. Sabin, M., A symmetric domain for a 6–sided patch, in *Mathematics of Surfaces IV*, Clarendon, 1994, 185–194.

55. Sabin, M., Transfinite surface interpolation, in *Mathematics of Surfaces VI*, Clarendon, 1996, 517–534.

56. Sabin, M., Two–sided patches suitable for inclusion in a B–Spline surface, in *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998, 409–416.

57. Sederberg, T. W., J. Zheng, D. Sewell and M. Sabin, Non-uniform recursive subdivision surfaces, *SIGGRAPH*, 1998, 387–394.

*58. Sone, J., K. Konno, H. Chiyokura, Surface interpolation using NURBS boundary Gregory patch for non-four-sided area, (in japanese), The Journal of the Information Processing Society of Japan **40** (1999), 710–718.

*59. Sone, J., K. Konno, H. Chiyokura, Surface interpolation of non-four-sided and concave area by NURBS boundary Gregory patch, *Curve and Surface Design: Saint-Malo 1999*, Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 2000, 389–398.

60. Varady, T., and C. M. Hoffmann, Vertex blending: problems and solutions, in *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998, 501–527.

61. Varady, T., and A. P. Rockwood, Geometric construction for setback vertex blends, Computer Aided Design **29** (1997), 413–425.

62. Varady, T., and T. Hermann, Best fit surface curvature at vertices of topologically irregular curve networks, in *Mathematics of Surfaces VI*, Clarendon, 1996, 411–428.

63. Varady, T. and A. P. Rockwood, Vertex blending based on the setback split, in *Mathematical Methods for Curves and Surfaces*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1995, 527–542.

64. Varady, T., Overlap patches: a new scheme for interpolating curve networks with n-sided regions, Comput. Aided Geom. Design, **8** (1989), 7–27.

65. Varady, T., Survey and new results in n-sided patch generation, in *Mathematcs of Surfaces II*, Oxford, 1987, 203–236.

*66. Vida, J., R. R. Martin and T. Varady, A survey of blending methods that use parametric surfaces, Computer Aided Design **26** (1994), 341–365.

67. Wallner, J. and H. Pottmann, Spline Orbifolds, in *Curves and Surfaces with Applications in CAGD* A. Le Méhauté, C. Rabut, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1997, 445–464.

68. Wang, X., F. Cheng and B. A. Barsky, Blending, smoothing, and interpolation of irregular mesh using n-sided Varady surfaces, preprint, 1997.

69. Wang, X. and F. Cheng, Smooth surface interpolation of irregular meshes, in Proc. 1997 International Conference on Manufacturing Automation (ICMA '97), Hong Kong, 1997, 1213–1218.

70. Warren, J., Creating multi–sided rational Bézier surfaces using base points, ACM Transactions on Graphics **11** (1992), 127–139.

71. Warren, J., Multi–sided rational surface patches with independent boundary control, in *Design and Application of Curves and Surfaces: Mathematics of Surfaces V*, Oxford, 1994, 281–294.

72. Zhao, Y., and A. Rockwood, A convolution approach to n-sided patches and vertex blending, in *Designing Fair Curves and Surfaces*, SIAM, 1994, 295–314.

73. Zheng, J. J., and A. A. Ball, Control point surfaces over non-four–sided areas, Comput. Aided Geom. Design **14** (1997), 808–821.

74. Zhou, J. and J. A. Gregory, Filling polygonal holes with bicubic patches, Comput. Aided Geom. Design **11** (1994), 391–410.

75. Zhou, J. and V. K. H. Lau, An arbitrary rectangular patch network scheme for free-form surfaces, in *Mathematics of Surfaces VI,* Oxford, 1996, 397–410.

76. Zhou, J. and V. K. H. Lau, Constructing GC^k continuous surfaces using bi–polynomial patches for a polygonal hole problem, in *Mathematics of Surfaces VII,* Information Geometers, 1997, 69–83.

*77. Zube, S., The n-sided toric patches and A-resultants, Preprint No. 11-99 Johannes Guteneberg Universitat Mainz.

78. Zube, S., n-sided toric patches and A-resultants, manuscript.

Pierre Malraison
Spatial Technology, Inc.
2425 55th Street, Suite 100
Boulder, CO 80301 USA
pierre_malraison@spatial.com

Approximated Planes in Parallel Coordinates

Tanya Matskewich, Alfred Inselberg and Michel Bercovier

Abstract. For the visualization of multivariate problems, a multidimensional system of *Parallel coordinates* is used which provides a one-to-one mapping between subsets of N-space and subsets of 2-space. A rigorous methodology for doing and *seeing* N-dimensional geometry emerges as well as several applications. Here an application to *Error Tolerancing* involving the visualization and characterization of "approximate coplanarity" is presented. The exact description of the neighborhood of an N-dimensional hyperplane in a parallel coordinate system is given.

§1. Introduction

The parallel coordinate system serves as a tool for visualization of multidimensional objects and multivariate relations. It was shown ([1,2]) that this representation gives a simple and constructive geometrical description for subsets of points which are strictly coplanar (i.e. belong to a common p-flat in N-dimensional case). This allows the visualization of coplanar points and the existence of linear dependencies between variables. It leads to numerous applications in different fields, and also practical applications involving finite error tolerancing. Here exact descriptions of approximated hyperplanes in the parallel coordinate system are given, providing a methodology for their visualization.

In the next section a brief review of previous results is given. It is followed in Section [3] by an exact mathematical formulation of the problem in the general case. Sections [4] and [5] contain some auxiliary lemmas which make the main result more intuitive, as well as the main result itself.

Although we have a complete and precise proof of the main result in the general case of "approximated" p-flats in N-dimensional space, lack of space prevents us from presenting it here.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 257–266.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

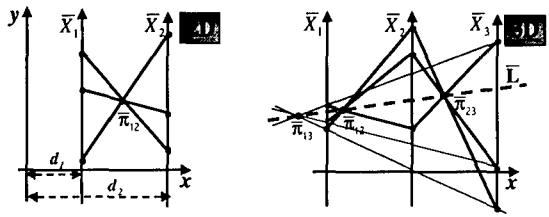


Fig. 1. Three collinear points in 2D (left) and 3D (right).

§2. Representation of Affine Subspaces in Parallel Coordinates

The parallel coordinate system is constructed in the following way: in the Euclidean plane $\mathbb{R}^2$ (or more precisely in the $2D$ projective plane $\mathbb{P}^2$) with xy-Cartesian coordinates, N copies of the axis y labeled $\bar{X}_1, \ldots, \bar{X}_N$ are placed equidistant (usually the distance between adjacent axes is taken as 1) and perpendicular to the x-axis. They are the axis of the Parallel Coordinate system.

A point with Euclidean coordinates $(p_1, \ldots, p_N)$ is represented by a polygonal line with N vertices $\bar{\pi}_i = (i-1, p_i)$, one on each axis. In this way, a $1--1$ correspondence between points in $\mathbb{R}^N$ and planar polygonal lines with vertices on the parallel axes is established.

In $2D$, a point is represented by a line (usually just the segment between the axes is shown). It can be easily proved that the lines representing points of a line $c_1x_1 + c_2x_2 = c_0$ (for $c_1 + c_2 \neq 0$) intersect at the point $\bar{\pi}_{12}$ with xy-coordinates $\left(\frac{c_2}{c_1+c_2}, \frac{c_0}{c_1+c_2}\right)$, or more generally at the point

$$\bar{\pi}_{12} = \left(\frac{d_1c_1 + d_2c_2}{c_1 + c_2}, \frac{c_0}{c_1 + c_2}\right), \tag{1}$$

where d_1 and d_2 are distances between axis y and $\bar{X}_1$, $\bar{X}_2$, respectively. (Lines with slope 1 are mapped onto the ideal points of projective plane, but in what follows we will not consider any "degenerate" cases). Hence a fundamental *point* $\leftrightarrow$ *line* duality is induced (see Fig. 1).

In $3D$, a line can be fully described by any pair from its three projections on coordinate planes. Each such projection is a line in 2D-space of the corresponding coordinates, and so can be represented in parallel coordinates exactly as was described above. Hence, if $c_i^{\{ij\}}x_i + c_j^{\{ij\}}x_j = c_0^{\{ij\}}$ - projection of the line on X_iX_j Euclidean plane $(i, j \in \{1, 2, 3\})$, then it is represented in the parallel coordinate system by the point $\bar{\pi}_{ij}$ whose coordinates may be computed from equation (1), and can be found geometrically as the intersection of corresponding lines. The three points $\bar{\pi}_{12}$, $\bar{\pi}_{23}$ and $\bar{\pi}_{13}$ are always collinear as a consequence of Desargue's Theorem, and any two of them represent the line in parallel coordinates. We denote by $\bar{L}$ the line on which the three points lie (see Fig. 1).

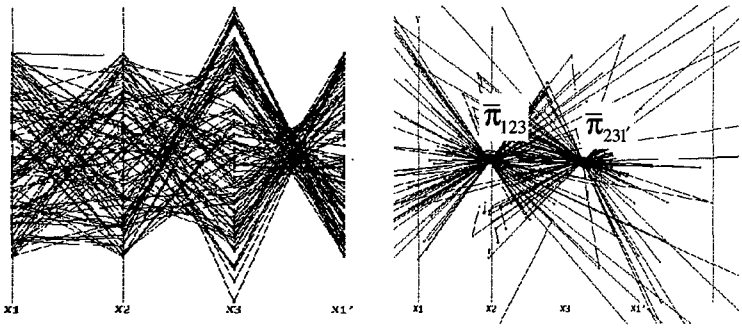


Fig. 2. Randomly sampled set of coplanar points in 3D and intersection of lines $\bar{L}$ showing coplanarity.

Let us now consider a plane $c_1x_1 + c_2x_2 + c_3x_3 = c_0$. Every pair of points belonging to this plane define a line $\bar{L}$ in parallel coordinates (which can be constructed geometrically from the representation of points themselves). It can be shown by direct computation that all such lines intersect at the common point with coordinates

$$\bar{\pi}_{123} = \left(\frac{c_1 + 2c_2}{c_1 + c_2 + c_3}, \frac{c_0}{c_1 + c_2 + c_3} \right). \tag{2}$$

This condition can be used to characterize coplanarity. Note, that one point is not sufficient in order to specify the plane. The solution is to introduce an additional axis $\bar{X}_1'$ placed after X_3 and at a unit distance from it, and consider the representation of points also in axes $(\bar{X}_2, \bar{X}_3, \bar{X}_1')$. This leads to the additional point $\bar{\pi}_{231'} = \left(\frac{c_2+2c_3+3c_1}{c_1+c_2+c_3}, \frac{c_0}{c_1+c_2+c_3} \right)$ (see Fig. 2).

This generalizes nicely to the N-dimensional case, and it can be shown that a representation of a hyperplane in parallel coordinates also can be recursively constructed by a simple geometric procedure, using affine subspaces of lower dimensions. A hyperplane is represented by $N-1$ indexed points; the "first" one has coordinates

$$\bar{\pi}_{12\ldots N} = \left(\frac{c_2 + 2c_3 + \ldots + (N-1)c_N}{c_1 + c_2 + \ldots + c_N}, \frac{c_0}{c_1 + c_2 + \ldots + c_N} \right), \tag{3}$$

and the others have very similar formulas.

A p-flat in N-dimensional case can be described by $N-p$ linearly independent equations, where each of them has the form $\sum_{k=1}^{p+1} c_{i_k} x_{i_k} = c_0$, and so corresponds to a hyperplane in axes $(\bar{X}_{i_1}, \bar{X}_{i_2}, \ldots, \bar{X}_{i_{p+1}})$. It follows that a p-flat is represented by $p(N-p)$ indexed points. The ensuing discussion is restricted to "approximated" hyperplanes. It is easy to show that the general case of an "approximate" p-flat in N-dimensional space can be reduced to the study of some "approximate" hyperplane.

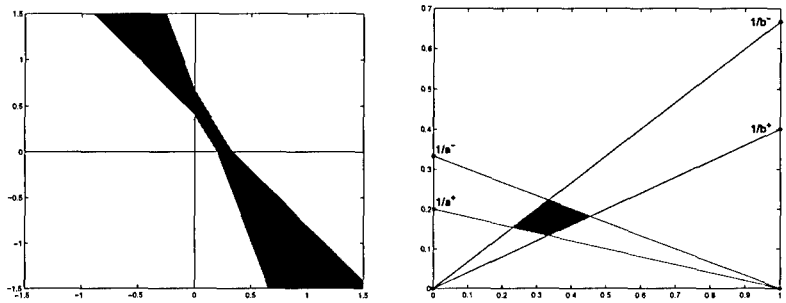


Fig. 3. Sampled proximate line $ax + by = 1$, $a \in [a^-, a^+]$, $b \in [b^-, b^+]$ in orthogonal and parallel coordinates.

§3. Exact Problem Definition

We will use the following definition of "similarity":

Definition 1. Proximate flats *are defined here as flats with proximate equations. An* approximate hyperplane *is defined as a set of hyperplanes given by equations*

$$c_1 x_1 + c_2 x_2 + \cdots + c_N x_N = c_0, \tag{4}$$

where every coefficient can vary: $c_i \in [c_i^-, c_i^+]$, $i = 0, \ldots, N$.

Such a slab of hyperplanes is extremely difficult to visualize in orthogonal coordinates, even for $3D$. Another problem is that even in the $2D$ case, line neighborhoods are unbounded in orthogonal coordinates, so neighborhoods for different lines always overlap. Fig. 3 shows that samples of proximate lines (in $2D$) form a cloud in the form of a very simple and nice convex quadrilateral.

In the N-dimensional case, a hyperplane is described by $N - 1$ points which means that in the approximated case, we will get $N - 1$ "clouds" of indexed points in parallel coordinates. In order to make things simpler, we will use the following

Assumption 2. *Free coefficients of equations are not allowed to vary, and are supposed to be identically equal to 1.*

(We have a complete analogue of the main result for the case when this assumption is not applied.)

Lemma 3. *It is sufficient to study only the range of the first indexed point of a hyperplane. That is, the following mathematical problem should be considered: find the range of the function* $f : \mathbb{R}^N \to \mathbb{R}^2$ *such that*

$$f(c_1, \ldots, c_N) = \left(\frac{\sum_{j=1}^{N} (j-1) c_j}{\sum_{j=1}^{N} c_j}, \frac{1}{\sum_{j=1}^{N} c_j} \right) \tag{5}$$

when $c_j \in [c_j^-, c_j^+]$ $(c_j^- < c_j^+, j = 1, \ldots, N)$.

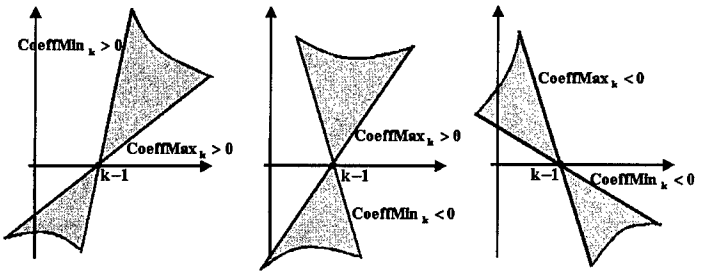


Fig. 4. Possible place for domain Ω depending on the signs of $CoeffMin_k$ and $CoeffMax_k$.

The notations used are:

- $x(c_1, \ldots, c_N)$ and $y(c_1, \ldots, c_N)$ - the first and the second coordinates of $f_N(c_1, \ldots, c_N)$ in $\mathbb{R}^2$ respectively;
- $B = [c_1^-, c_1^+] \times \ldots \times [c_N^-, c_N^+]$ - box in the space of coefficients;
- $\Omega = f(B)$ - image of B in the parallel coordinate system.

We now show why Lemma 3 holds in the $3D$ case. Here studying the range of $\bar{\pi}_{123}$ (equation (2)) is sufficient because $\bar{\pi}_{231'}$ can be rewritten in the form

$$\pi_{231'} = (1,0) + \left(\frac{c_3 + 2c_1}{c_1 + c_2 + c_3}, \frac{1}{c_1 + c_2 + c_3} \right)$$

This implies that the equation of $\bar{\pi}_{231'}$ can be obtained from the one of $\bar{\pi}_{123}$ by shift and cyclical change of parameters $c_1 \to c_2$, $c_2 \to c_3$, $c_3 \to c_1$.

For the general case, this "reduction" lemma can be proved using simple combinatorics, and it can be shown that in order to describe the range of one of the indexed points corresponding to a p-flat in N-dimensional space, it is always sufficient to consider f_{2N} with some coefficients identically equal to zero.

§4. Some Notes on the Domain Ω

To get an intuitive feel about the structure of the domain Ω, we consider the possible location of Ω and what it looks like. Note that for every $k = 1, \ldots, N$ the following representation takes place:

$$x(c_1, \ldots, c_N) = k - 1 + \frac{\sum_{j=1}^{N} (j-k) c_j}{\sum_{j=1}^{N} c_j} = k - 1 + Coeff_k \; y(c_1, \ldots, c_N), \quad (6)$$

where $Coeff_k = \sum_{j=1}^{k-1} \underbrace{(j-k)}_{<0} c_j + \sum_{j=k+1}^{N} \underbrace{(j-k)}_{>0} c_j$ does not depend on c_k.

If only c_k varies, while the other coefficients are fixed, then (x, y) lie on the straight line which passes through the point $(k-1, 0)$ and has slope $1/Coeff_k$. Here $CoeffMin_k \le Coeff_k(c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_N) \le CoeffMax_k$

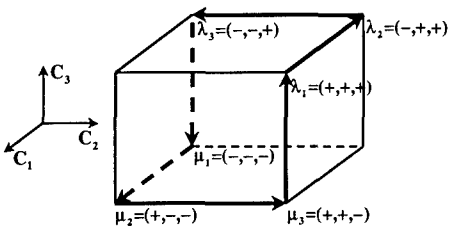


Fig. 5. Important vertices and edges in the space of parameters.

for every choice of $(c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_N)$, where

$$\begin{aligned} \mathit{CoeffMin}_k &= \mathit{Coeff}_k(c_1^+, \ldots, c_{k-1}^+, c_{k+1}^-, \ldots, c_N^-), \\ \mathit{CoeffMax}_k &= \mathit{Coeff}_k(c_1^-, \ldots, c_{k-1}^-, c_{k+1}^+, \ldots, c_N^+). \end{aligned} \tag{7}$$

The domain Ω lies between lines with maximal and minimal slopes. More precisely

Lemma 4. *Ω lies above the line corresponding to $\mathit{CoeffMax}_k$ iff*

$\mathit{CoeffMax}_k > 0$ (otherwise it lies below the line). Ω lies above the line corresponding to $\mathit{CoeffMin}_k$ iff $\mathit{CoeffMin}_k < 0$ (see Fig. 4).

Let us now introduce the following notations for some important vertices and edges of box B. There are $2N$ (among 2^N) important vertices:

$$\begin{aligned} \lambda_k &= (c_1^+, c_2^+, \ldots, c_{k-1}^+, c_k^-, c_{k+1}^-, \ldots, c_N^-), \\ \mu_k &= (c_1^-, c_2^-, \ldots, c_{k-1}^-, c_k^+, c_{k+1}^+, \ldots, c_N^+). \end{aligned} \tag{8}$$

for $k = 1, \ldots, N$ and $2N$ important edges connecting these vertices (see Fig. 5):

$$\begin{aligned} \alpha_k(c_k) &= (c_1^-, \ldots, c_{k-1}^-, c_k, c_{k+1}^+, \ldots, c_N^+), \quad c_k \in [c_k^+, c_k^-], \\ \beta_k(c_k) &= (c_1^+, \ldots, c_{k-1}^+, c_k, c_{k+1}^-, \ldots, c_N^-), \quad c_k \in [c_k^-, c_k^+]. \end{aligned} \tag{9}$$

Edge α_k connects vertices λ_k and λ_{k+1}, edge β_k - vertices μ_k and μ_{k+1}. (Here $\lambda_{N+1} = \mu_1$ and $\mu_{N+1} = \lambda_1$).

We also introduce the notation

$$\mathit{sum}(c_1, \ldots, c_N) = c_1 + \cdots + c_N \tag{10}$$

which will be useful in what follows.

As explained above, it is clear that α_k is mapped onto the boundary line of Ω corresponding to $\mathit{CoeffMax}_k$, and β_k is mapped onto the boundary line corresponding to $\mathit{CoeffMin}_k$. More precisely, we have

Lemma 5. *The image of α_k is the segment between $f_N(\lambda_k)$ and $f_N(\lambda_{k+1})$ if $y = 1/\mathit{sum}(\alpha(c_k))$ does not change sign while $c_k \in [c_k^+, c_k^-]$, and the complement of the straight line to this segment otherwise. In other words, $f_N(\alpha_k)$*

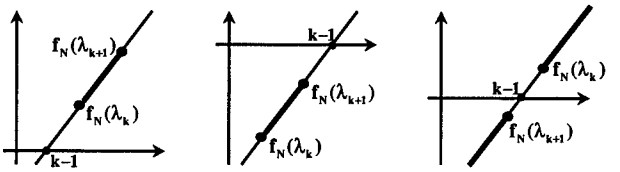


Fig. 6. Image of α_k depending on the signs of $y(\lambda_k)$ and $y(\lambda_{k+1})$.

is a segment if plane $c_1 + c_2 + \cdots + c_N = 0$ *does not intersect edge* α_k *of box* B*, and a complement of the segment otherwise (see Fig. 6).*

Of course, the analogous statement holds for the image of β_k. In what follows, we will usually formulate only statements for λ_k and α_k, and omit the analogous ones for μ_k and β_k.

Conclusion 6. *$f_N(\lambda_k)$ and $f_N(\lambda_{k+1})$ are connected by segment iff they lie in the same (upper or lower) half-plane, i.e. if $y(\lambda_k)$ and $y(\lambda_{k+1})$ have the same signs.*

Note further, that every one of the points $f_N(\lambda_k)$ ($f_N(\mu_k)$) is a point of the concatenation of two boundary segments corresponding to $CoeffMax_{k-1}$ and $CoeffMax_k$ ($CoeffMin_{k-1}$ and $CoeffMin_k$, respectively).

In order to make this precise and to assure that all boundary can be described in this manner, the following theorem was proved.

Theorem 7. *$f_N(c_1, \ldots, c_N)$ belongs to the boundary of domain Ω iff there exists $k = 1, \ldots, N$ such that $(c_1, \ldots, c_N) = \alpha_k(c_k)$ or $(c_1, \ldots, c_N) = \beta_k(c_k)$ for $c_k \in [c_k^-, c_k^+]$.*

Again the proof (which is relatively long) is omitted the proof uses the "topological" notion of point neighborhood, boundary etc.

Conclusion 8. *In order to describe the boundary of the domain Ω, it suffices to move along the following contour in the space of coefficients*

$$\begin{array}{ccccccccccc} & \nearrow & \lambda_2 & \rightarrow & \lambda_3 & \rightarrow & \ldots & \rightarrow & \lambda_N & \searrow & \\ \lambda_1 & & & & & & & & & & \mu_1 \\ & \nwarrow & \mu_N & \leftarrow & \mu_{N-1} & \leftarrow & \ldots & \leftarrow & \mu_2 & \swarrow & \end{array} \tag{11}$$

Note that independently of the specific values of c_j^- and c_j^+ ($j = 1, \ldots, N$), always only $2N$ (among 2^N) definite vertices and edges of B and in definite order participate in the boundary of domain Ω.

It remains to "fill in" the boundary of Ω with Ω itself. Before we formulate the main result, let us study an additional property of domain Ω and its boundary.

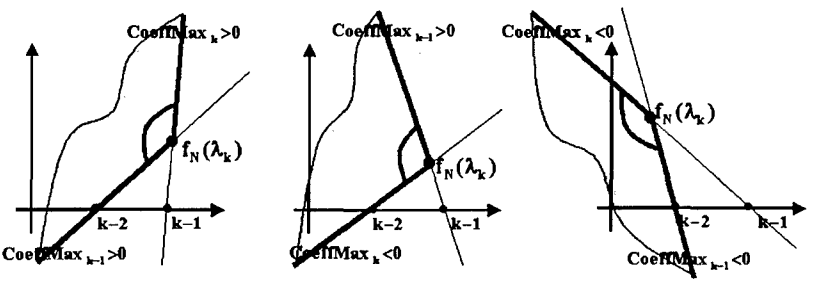


Fig. 7. Convexity of vertex $f_N(\lambda_k)$.

Lemma 9. *The domain Ω has only convex boundary vertices.*

Indeed, it was shown above that a boundary vertex of Ω of the form $f_N(\lambda_k)$ is the intersection of the lines with slopes $1/CoeffMax_{k-1}$ and $1/CoeffMax_k$ and passing through points $(k-2,0)$ and $(k-1,0)$ respectively. Note that the following equation holds:

$$CoeffMax_{k-1} - CoeffMax_k = sum(\lambda_k) = 1/y(\lambda_k) \tag{12}$$

Assume for example that $f_N(\lambda_k)$ lies in upper half-plane, i.e., that $y(\lambda_k) > 0$ (the analogous consideration can be done for the lower hyperplane). Then $CoeffMax_{k-1} > CoeffMax_k$, and using Lemma 4 we get that in any one of three possible cases (see Fig. 7) this vertex is convex.

§5. The Main Result - Description of the Domain Ω

Theorem 10. *The domain Ω has one of two possible forms depending on whether $sum(\lambda_1) = c_1^+ + c_2^+ + \cdots + c_N^+$ and $sum(\mu_1) = c_1^- + c_2^- + \cdots + c_N^-$ has the same sign or not.*

Note that this condition is equivalent to the condition that the plane $c_1 + c_2 + \cdots + c_N = 0$ intersects the box B in the space of coefficients.

Case 1. If $sum(\lambda_1)$ and $sum(\mu_1)$ have the same sign, then Ω is a convex bounded polygon inside the contour (11), (see Fig. 8 and 9 - left parts. In the figures we will write $\bar{\lambda}_k$ instead of $f_N(\lambda_k)$ in order to make the figures clearer and more compact).

Indeed, let us assume that $sum(\mu_1) > 0$. Then $y(\lambda_k) > 0$ and $y(\mu_k) > 0$ for every $k = 1, \ldots, N$ (all vertices of Ω lie in the upper half-plane). According to Conclusion 6, the boundary of Ω in this case consists of segments which form a convex (Lemma 9) bounded polygon.

Case 2. If $sum(\mu_1) < 0$ and $sum(\lambda_1) > 0$, then $sum()$ changes its sign exactly once when upper or lower chain of the contour (11) is traversed, say at the segment $\alpha_k = [\lambda_k, \lambda_{k+1}]$ at the upper chain and segment $\beta_p = [\mu_p, \mu_{p+1}]$ at the lower chain. Then the domain Ω is a union of two convex unbounded

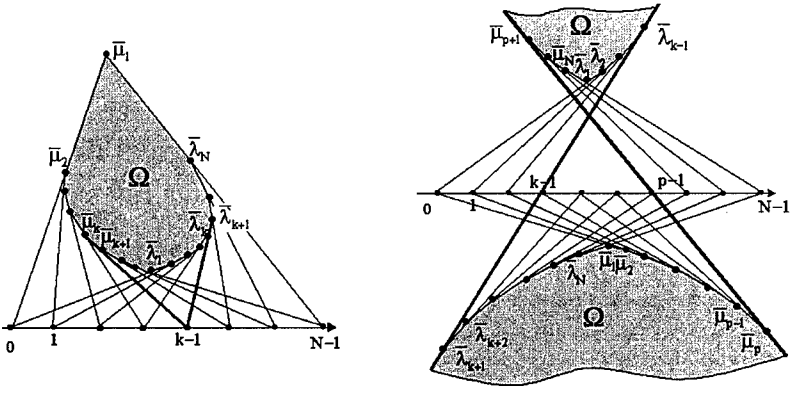


Fig. 8. The domain Ω in the general case.

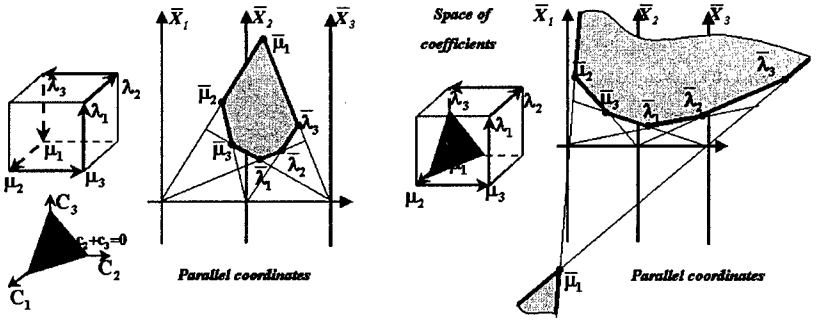


Fig. 9. An example of the domain Ω in the $3D$ case.

polygons. The first is inside the part of the contour that belongs to the upper half-plane, and the second is inside the part of the contour that belongs to the lower half-plane.

Upper and lower parts are bounded by infinite rays with the same slopes (corresponding to $CoeffMax_k$ and $CoeffMin_p$), i.e. it is a *convex bounded polygon in the projective plane* (see Fig. 8 and 9 - right parts).

§6. Example of an Affine Subspace of Lower Dimension

We now show how the general result can be applied to the construction of p-flats of lower dimensions, for the "approximate" line in $3D$ case. For example, if we would like to describe the range of the indexed point $\bar{\pi}_{13} = \left(\frac{2c_3^{\{13\}}}{c_1^{\{13\}}+c_3^{\{13\}}}, \frac{1}{c_1^{\{13\}}+c_3^{\{13\}}}\right)$ which enters in the representation of the line, then we can reduce it to the consideration of f_3 by putting $c_2 \in [0,0]$. We get "void" connections instead of edges α_2 and β_2, and finally get a convex quadrilateral instead of convex hexagon (see Fig. 10).

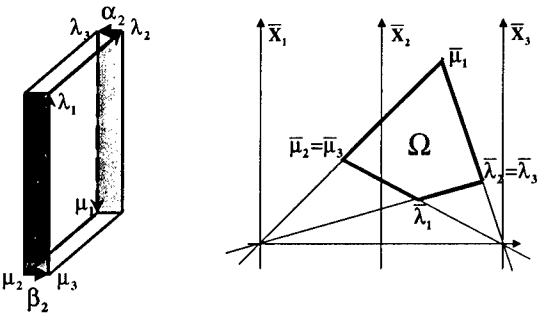


Fig. 10. Range of $\bar{\pi}_{12}$ - the first indexed point corresponding to line in $3D$ case.

References

1. Eickemeyer, J., Visualizing p-flats in N-spave using Parallel Coordinates, Ph.D. Thesis UCLA, 1992.
2. Inselberg, A., and B. Dimsdale, Multidimensional lines I: Representation, SIAM J. of Applied Math., **54-2** (1994), 559-577.

Tanya Matskewich, Michel Bercovier
The Hebrew University of Jerusalem, Israel
fisa@cs.huji.ac.il
berco@cs.huji.ac.il

Alfred Inselberg
A. Inselberg
School of Mathematical Sciences
Tel Aviv University, Tel Aviv 61396, Israel
aiisreal@cs.tau.ac.il

Applications of Sphere Geometry in Canal Surface Design

Christoph Mäurer

Abstract. Classical models of sphere geometry facilitate an efficient description of canal surfaces. Using the cyclographic model of Laguerre geometry, an elementary characterization of continuity, bisectors, control point-, control sphere- and implicit representations of canal surfaces is presented. In addition, canal surfaces generated with the aid of Minkowski Pythagorean hodograph curves are investigated.

§1. Introduction

A canal surface C in $\mathbb{R}^3$ is defined as the envelope surface of a moving sphere $S(t)$ with center $\boldsymbol{m}(t) := (m_1(t), m_2(t), m_3(t))$ and radius function $r(t)$. The moving sphere can be described with the implicit equation

$$F(\boldsymbol{x}, t) = \|\boldsymbol{x} - \boldsymbol{m}(t)\|^2 - r^2(t) = 0. \tag{1}$$

The envelope condition

$$\frac{\partial F}{\partial t} = [\boldsymbol{x} - \boldsymbol{m}(t)] \cdot \dot{\boldsymbol{m}} + r(t)\dot{r}(t) = 0 \tag{2}$$

describes a moving plane, which intersects $S(t)$ in the characteristic circles of the canal surface. It is a natural approach to use models of classical sphere geometry to study canal surfaces. Well-known models have been investigated in Möbius geometry, Laguerre geometry and Lie geometry. For an overview on sphere geometry, the reader may consult [2]. Papers which handle sphere geometry in the CAGD context are [7,8,10,11,12].

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 267–276.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

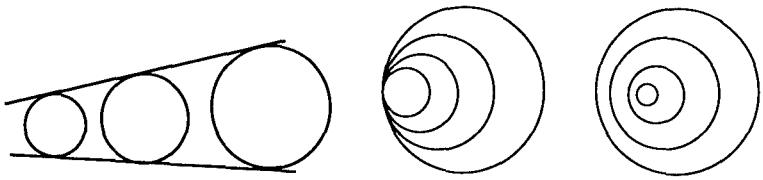


Fig. 1. Hyperbolic, parabolic and elliptic sphere families.

§2. Fundamentals of Laguerre Geometry

In the cyclographic model $\mathbb{R}^{3,1}$ of Laguerre geometry, a sphere S with center $\boldsymbol{m} = (m_1, m_2, m_3)$ and radius r is described as a point $\boldsymbol{s} = (m_1, m_2, m_3, r)^T \in \mathbb{R}^{3,1}$. It is an intuitive model, since the space of points (spheres with vanishing radius) is embedded in the cyclographic model as the hyperplane $x_4 = 0$. The absolute quadric Ω, which reads in homogeneous coordinates $(\underline{x}_i/\underline{x}_0 := x_i)$ as $\Omega : \underline{x}_0 = \underline{x}_1^2 + \underline{x}_2^2 + \underline{x}_3^2 - \underline{x}_4^2 = 0$ plays an important role. It defines a pseudo Euclidean (pe) metric in $\mathbb{R}^{3,1}$ via the scalar product

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle_{pe} := a_1 b_1 + a_2 b_2 + a_3 b_3 - a_4 b_4. \tag{3}$$

The pe distance $\|\boldsymbol{a} - \boldsymbol{b}\|_{pe} := \sqrt{\langle \boldsymbol{a} - \boldsymbol{b}, \boldsymbol{a} - \boldsymbol{b} \rangle_{pe}}$ of two points $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^{3,1}$ measures the tangential distance of two spheres A, B. First we study the most simple family of spheres described as a line L with direction vector $\boldsymbol{a}$ in $\mathbb{R}^{3,1}$. There are three cases: If $\langle \boldsymbol{a}, \boldsymbol{a} \rangle_{pe} > 0$, then L is called a hyperbolic line corresponding to a family of spheres whose envelope surface is a right circular cone (Fig. 2, left). If $\langle \boldsymbol{a}, \boldsymbol{a} \rangle_{pe} = 0$, then L is called a parabolic or isotropic line belonging to spheres in oriented contact. If $\langle \boldsymbol{a}, \boldsymbol{a} \rangle_{pe} < 0$, then L is called a elliptic line and two arbitrary spheres of such a family do not have a common tangent plane. These three cases of sphere families are plotted in Fig. 1 (but to simplify matters in $\mathbb{R}^2$ instead of $\mathbb{R}^3$). A canal surface is completely determined by the set of its tangent cones. They belong to hyperbolic tangent vectors of a curve $\boldsymbol{c}(t) \in \mathbb{R}^{3,1}$. Vice versa, the set of hyperbolic tangent vectors of $\boldsymbol{c}(t)$ corresponds to the set of tangent cones of a canal surface along the characteristic circles. For elliptic tangent vectors, there does not exist a real tangent cone. Therefore, that case has to be avoided in the specification of canal surfaces. Discrete parameter values t_0 with parabolic tangent vectors ($\|\dot{\boldsymbol{c}}(t_0)\|_{pe} = 0$) are tolerated. At these parameter values, the canal surface is closed (Fig. 2 right).

Corollary 1. *Any real G^1-continuous canal surface C in $\mathbb{R}^3$ can be described as a G^1-continuous curve $\boldsymbol{c}(t) \in \mathbb{R}^{3,1}$ with hyperbolic tangent vectors.*

A Laguerre transformation τ in $\mathbb{R}^{3,1}$ is an affine mapping $\boldsymbol{x} \mapsto \lambda \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$, $0 < \lambda \in \mathbb{R}$ which preserves Ω. It turns out that this condition is fulfilled iff $\boldsymbol{A}$ is an orthogonal matrix with respect to the pe metric, i.e.

$$\boldsymbol{A}^T \boldsymbol{E}_{pe} \boldsymbol{A} = \boldsymbol{E}_{pe} := diag(1, 1, 1, -1). \tag{4}$$

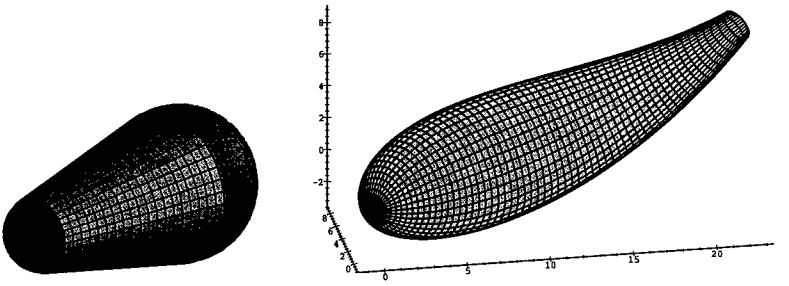


Fig. 2. Left: Tangent cone of two spheres; right: Effect of parabolic tangent vector.

Therefore, Laguerre transformations are pe similarities in $\mathbb{R}^{3,1}$. Since the Euclidean space $\mathbb{R}^3$ is embedded in $\mathbb{R}^{3,1}$, any Euclidean translation or rotation is a Laguerre transformation. The offset operation with offset distance d is described as a translation τ_d in the x_4-direction $(\lambda = 1, \boldsymbol{A} = \boldsymbol{E}, \boldsymbol{b} = (0,0,0,d))$. A further example is the pe rotation around a fixed 2-plane. If the x_2, x_3-plane is fixed, the pe rotation is represented by

$$\tau_r : \begin{pmatrix} x_1 \\ x_4 \end{pmatrix} \mapsto \begin{pmatrix} \cosh\alpha & \sinh\alpha \\ \sinh\alpha & \cosh\alpha \end{pmatrix} \begin{pmatrix} x_1 \\ x_4 \end{pmatrix}. \tag{5}$$

Any Laguerre transformation with $\lambda = 1$ preserves the tangential distance. The following examples show how Laguerre transformations act on simple curves of $\mathbb{R}^{3,1}$.

Example 1. *Consider a straight line $L \in \mathbb{R}^3$. The image line under the offset mapping τ_d describes a cylinder. An additional pe rotation $\tau_r \circ \tau_d(L)$ yields a hyperbolic line corresponding to a right circular cone. Vice versa, any hyperbolic line in $\mathbb{R}^{3,1}$ can be mapped via an appropriate Laguerre transformation to L.*

The line in Example 1 does not change its type under the mappings τ_r and τ_d. Since the sign of $\langle \boldsymbol{a}, \boldsymbol{a} \rangle_{pe}$ is not modified by any Laguerre transformation, we obtain:

Corollary 2. *The type (hyperbolic, parabolic, elliptic) of straight lines in the cyclographic model $\mathbb{R}^{3,1}$ is invariant under Laguerre transformations.*

Example 2. *Consider a Euclidean circle $K \in \mathbb{R}^3$. Its image $\tau_d(K)$ describes a* torus, *and the pe circle $\tau_r \circ \tau_d(K)$ corresponds to a* Dupin cyclide. *Any Dupin cyclide can be interpreted as a circle in $\mathbb{R}^{3,1}$ with respect to the pseudo Euclidean metric. For more details and a proof of this fact, see [7].*

The pe circles might be utilized to describe G^2-continuity between canal surfaces: The osculating pe circle of a curve $\boldsymbol{c}(t) \in \mathbb{R}^{3,1}$ belongs to the osculating cyclide of a canal surface C, which specifies the curvature behaviour of C. A technique to compute osculating circles of space curves in the Euclidean and non-Euclidean space is given in [1].

§3. Isotropic Hypersurfaces

Definition 1. *Let $C \subset \mathbb{R}^3$ be an oriented C^1-continuous canal surface, which belongs to the curve $\boldsymbol{c} \in \mathbb{R}^{3,1}$. The* isotropic hypersurface $\Gamma(\boldsymbol{c}) \subset \mathbb{R}^{3,1}$ *is the union of all points in $\mathbb{R}^{3,1}$ corresponding to all oriented tangent spheres of C.*

For a fixed surface point of C, all oriented tangent spheres are represented as isotropic lines in $\mathbb{R}^{3,1}$. Therefore, $\Gamma(\boldsymbol{c})$ is formed by a two-parametric family of isotropic lines. Applying Corollary 2 to the definition of isotropic hypersurfaces, we obtain:

Proposition 1. *Isotropic hypersurfaces are invariant under Laguerre transformations $\tau(\Gamma(c)) = \Gamma(\tau(c))$.*

The simplest example is the isotropic hypersurface of a sphere S specified by a point $\boldsymbol{s} \in \mathbb{R}^{3,1}$. It is a hypercone $\Gamma(s) = \langle \boldsymbol{x} - \boldsymbol{s}, \boldsymbol{x} - \boldsymbol{s} \rangle_{pe} = 0$. Now consider a canal surface C characterized by a curve $\boldsymbol{c}(t) \in \mathbb{R}^{3,1}$ with hyperbolic tangent vector $\dot{\boldsymbol{c}}(t)$. As shown in [7], its isotropic hypersurface can be achieved in a two-step procedure: First intersect the pe polar plane $\dot{\boldsymbol{c}}_{pe}^{\perp}(t)$ with the absolute quadric Ω (the pe polar plane $\boldsymbol{a}_{pe}^{\perp}$ of a vector $\boldsymbol{a} \in \mathbb{R}^{3,1}$ is defined as $\boldsymbol{a}_{pe}^{\perp} : \boldsymbol{a}^T \boldsymbol{E}_{pe} \boldsymbol{x} = 0$). Then join the resulting family of conics $\boldsymbol{k}(t,s)$ and the curve points $\boldsymbol{c}(t)$ with straight lines. Or as a formula:

$$\Gamma(\boldsymbol{c}) = \cup_t \, \boldsymbol{c}(t) \star \{\dot{\boldsymbol{c}}_{pe}^{\perp}(t) \cap \Omega\},$$

where $A \star B$ denotes all straight lines joining A and B.

Isotropic hypersurfaces have several properties which can be applied in the context of geometric design:

Proposition 2.

1) *The intersection of the isotropic hypersurface $\Gamma(\boldsymbol{c})$ with $\mathbb{R}^3$ generates the canal surface C itself: $\Gamma(\boldsymbol{c}) \cap \mathbb{R}^3 = C$.*
2) *The intersection of $\tau_d(\Gamma(c))$ with $\mathbb{R}^3$ gives the offset surface of C.*
3) *Consider two canal surfaces $C_1, C_2 \in \mathbb{R}^3$ with corresponding curves $\boldsymbol{c}_1(t), \boldsymbol{c}_2(t) \in \mathbb{R}^{3,1}$. These canal surfaces touch each other if $\boldsymbol{c}_2$ is lying on $\Gamma(\boldsymbol{c}_1)$ (resp. if $\boldsymbol{c}_1$ is lying on $\Gamma(\boldsymbol{c}_2)$).*
4) *The bisector surface of C_1 and C_2 is formed by the orthogonal projection of $\{\Gamma(\boldsymbol{c}_1) \cap \Gamma(\boldsymbol{c}_2)\}$ to $\mathbb{R}^3$.*

Proof: *1) The tangent spheres with radius zero of a canal surface are precisely its surface points. They are contained in the hyperplane $\mathbb{R}^3$ of the cyclographic model. 2) follows from the offset property of τ_d and Proposition 1. 3) follows immediately from Definition 1. To prove 4), note that the bisector is characterized as the center of spheres which have oriented contact to C_1 and C_2. In $\mathbb{R}^{3,1}$ these spheres are characterized by $\{\Gamma(\boldsymbol{c}_1) \cap \Gamma(\boldsymbol{c}_2)\}$.* □

We complete this section with few examples which make use of Laguerre transformations and isotropic hypersurfaces.

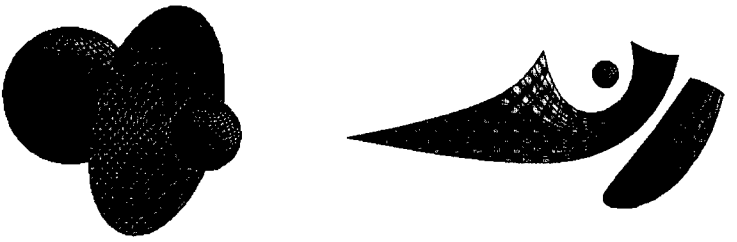


Fig. 3. Bisector of 2 spheres, resp. of sphere and canal surface.

Example 3. *Using the previous proposition and Example 2, implicit and parametric representation of cyclides C can be derived easily. With a Laguerre transformation τ, the corresponding pe circle can be mapped to a simple normal form $\underline{x}_3 = \underline{x}_4 = \underline{x}_1^2 + \underline{x}_2^2 - \underline{x}_0^2 = 0$ of a image cyclide $\tilde{C}$. Its isotropic hypersurface reads in implicit form as*

$$\Gamma(\tilde{c}) : (\underline{x}_0^2 + \underline{x}_1^2 + \underline{x}_2^2 + \underline{x}_3^2 - \underline{x}_4^2)^2 - 4\underline{x}_0^2(\underline{x}_1^2 + \underline{x}_2^2) = 0,$$

and in parametric form as

$$(\underline{x}_0(u,v,w), \underline{x}_1(u,v,w), \underline{x}_2(u,v,w), \underline{x}_3(u,v,w), \underline{x}_4(u,v,w)) =$$
$$([1-w][1+v^2] + w[1-u^2], [1-w][1-v^2], 2[1-w]v, 2wu, -w[1+u^2]).$$

The implicit and parametric representation of the original cyclide C can be obtained from $\{\tau^{-1}(\Gamma(\tilde{c}))\} \cap \mathbb{R}^3$. The intersection procedure is straightforward, since τ is a linear mapping. In the parametric case, the parameter w can be eliminated easily, because it occurs linearly in the parameter representation.

Example 4. *Consider two spheres S_1, S_2 specified by points $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathbb{R}^{3,1}$. The intersection $\Gamma(\boldsymbol{s}_1) \cap \Gamma(\boldsymbol{s}_2)$ of their isotropic hypersurfaces is contained in the hyperplane $H_{12} : \langle \boldsymbol{x} - (\boldsymbol{s}_1 + \boldsymbol{s}_2)/2, \boldsymbol{s}_2 - \boldsymbol{s}_1 \rangle_{pe} = 0$. Thus, the bisector of S_1 and S_2 is the orthogonal projection of $H_{12} \cap \Gamma(\boldsymbol{s}_1)$ onto $\mathbb{R}^3$. Since $\Gamma(\boldsymbol{s}_1)$ is a hypercone, the bisector surface is a quadric surface. Fig. 3 shows two spheres and a part of their bisector surface (hyperboloid of two sheets).*

Example 5. *A similar technique allows the computation of the bisector of a sphere S and a canal surface C (specified by $\boldsymbol{s}$ and $\boldsymbol{c}(t) \in \mathbb{R}^{3,1}$). Now, $H_{12}(t) : \langle \boldsymbol{x} - (\boldsymbol{s} + \boldsymbol{c}(t))/2, \boldsymbol{c}(t) - \boldsymbol{s} \rangle_{pe} = 0$ depends on t. The bisector surface is obtained by the orthogonal projection of $H_{12} \cap \Gamma(\boldsymbol{c})$ onto $\mathbb{R}^3$. For a rational canal surface, $\Gamma(\boldsymbol{c})$ is a rational ruled hypersurface with parameter representation $\boldsymbol{g}(t,u,v)$ in $\mathbb{R}^{3,1}$ which is linear in v. After the intersection and projection operation, we obtain a rational bisector surface $\boldsymbol{b}(t,u)$. Fig. 3 depicts the bisector $\boldsymbol{b}(t,u)$ of a sphere and a canal surface $\boldsymbol{c}(t,u)$ with cubic spine curve and rational parametric degree (7,2). $\boldsymbol{b}(t,u)$ has parametric degree (10,2).*

Rational bisectors of spheres and PN surfaces are studied in [3]. Further applications of the concept of isotropic hypersurfaces can be found in [8] (tangent plane property) and [12] (offset property).

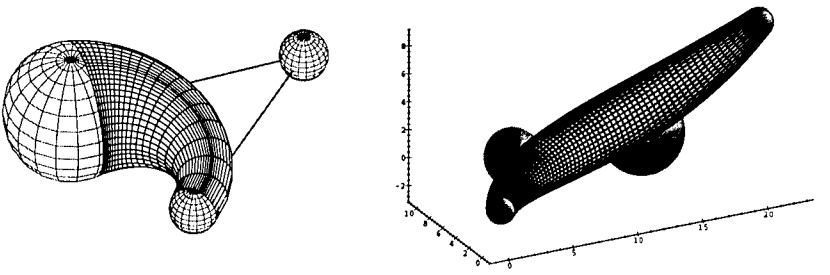


Fig. 4. Cyclide and cubic canal surface with control spheres.

§4. Control Sphere Representation

Consider a Bézier curve

$$\boldsymbol{x}(t) = \sum_{k=0}^{n} \boldsymbol{s}_k B_k^n(t) \tag{6}$$

of degree n in the cyclographic model with control points $\boldsymbol{s}_k \in \mathbb{R}^{3,1}$. They correspond to control spheres of the canal surface C described by $\boldsymbol{x}(t)$. Well-known properties of the Bernstein-Bézier technique can be applied directly to the Laguerre geometric approach (this idea was introduced in [12]):

Proposition 3.

1) *The de Casteljau algorithm enables a stable computation of the moving spheres of a canal surface and can be used to subdivide it into two parts.*
2) *The control spheres S_0 and S_1 (S_n and S_{n-1}) generate the tangent cone of the canal surface at $t = 0$ ($t = 1$).*
3) *A (rational) canal surface (with positive weights) lies in the convex hull of its control spheres.*

The control structure is useful from the designers point of view because the influence of moving the control spheres or changing their radius or weights is analogous to the well-known curve case. For example, Fig. 5 views the influence of modifying one weight of a canal surface with cubic spine curve: The weight β_1 is increased from 1 (left) to 10 (right). All other weights satisfy $\beta_i = 1$.

We have seen some advantages of using the control sphere representation of canal surfaces. However, often it is necessary to know a (rational) tensor-product representation. Therefore, one has to analyze the correlation between control spheres and classical control points of a canal surface. For the cyclide case there exists a simple geometric relation, which is described in [8]. In the case of an arbitrary canal surface, the problem was solved by Pottmann and Peternell [11]. They proved the surprising result that any canal surface with rational spine curve $\boldsymbol{m}(t)$ and rational radius function $r(t)$ is rational. Furthermore, they proved that the problem of finding the rational tensor-product representation can be reduced to the problem of finding two rational

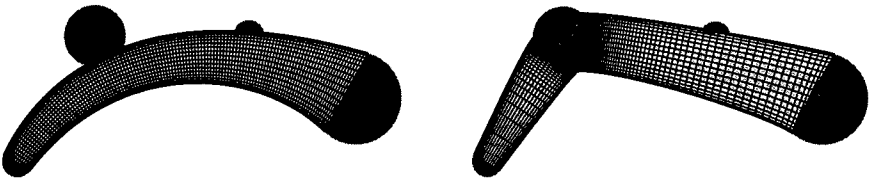


Fig. 5. Influence of different weights.

functions $\rho_1(t)$ and $\rho_2(t)$ which satisfy

$$\rho_1^2 + \rho_2^2 = \rho := \dot{m}_1^2 + \dot{m}_2^2 + \dot{m}_3^2 - \dot{r}^2. \tag{7}$$

Although the existence of solutions of (7) can be proved via factorization over the complex field, it is not trivial to find the (non-unique) solutions. There are two different options to perform the conversion from control spheres to control points:

1) Use arbitrary rational curves $\boldsymbol{x}(t) \in \mathbb{R}^{3,1}$ to describe canal surfaces. Then it is impossible to solve (7) exactly and numerical methods are required to compute the control points from the control spheres of the dedicated canal surface.
2) Use rational curves $\boldsymbol{x}(t) \in \mathbb{R}^{3,1}$ which have the property that $\rho(t)$ is a square in the polynomial ring $\mathbb{R}[t]$. Then an exact conversion from control spheres to control points can be realized using the algorithm of Pottmann and Peternell [11].

In this paper we will follow the second option. It results in the concept of Minkowski Pythagorean Hodograph curves.

§5. Minkowski Pythagorean Hodograph Curves in $\mathbb{R}^{3,1}$

Planar and spatial Pythagorean hodograph curves have been introduced by Farouki and Sakkalis [4,5]. Recently Moon [9] has generalized this class of curves and investigated the Minkowski Pythagorean hodograph (MPH) curves of $\mathbb{R}^{2,1}$. Here we need a further generalization: the MPH curves of $\mathbb{R}^{3,1}$:

Definition 2. *A polynomial (rational) curve* $\boldsymbol{x}(t) = (x_1(t), x_2(t), x_3(t), x_4(t))$ *in* $\mathbb{R}^{3,1}$ *is a* Minkowski Pythagorean hodograph *(MPH) curve if there exists a real polynomial (rational function)* $\sigma(t)$ *which satisfies*

$$\dot{x}_1^2(t) + \dot{x}_2^2(t) + \dot{x}_3^2(t) - \dot{x}_4^2(t) = \sigma^2(t). \tag{8}$$

As described before, for canal surfaces described via MPH curves of $\mathbb{R}^{3,1}$, an exact conversion from control spheres to control points can be realized. We now present some additional important properties of MPH curves.

Proposition 4.

1) *The tangent vector of a MPH curve is hyperbolic except a finite number of parabolic tangent vectors, i.e., the corresponding canal surface is always real.*
2) *The MPH property is invariant under Laguerre transformations.*
3) *PH space curves are exactly those MPH curves, which hold* $x_4 = 0$

Pe circles are examples for rational MPH curves of degree 2. In order to create polynomial MPH curves, equation (8) could be solved in a polynomial ring $\mathbb{R}[t]$. A possible solution (determined via stereographic projection) is

$$\begin{aligned}&(\dot{x}_1(t),\ \dot{x}_2(t),\ \dot{x}_3(t),\ \dot{x}_4(t),\ \sigma(t)) = \\ &(2u_0u_1,\ 2u_0u_2,\ -u_0^2+u_1^2+u_2^2-u_3^2,\ 2u_0u_3,\ u_0^2+u_1^2+u_2^2-u_3^2).\end{aligned} \tag{9}$$

The polynomials $\boldsymbol{u}(t) := (u_0(t), u_1(t), u_2(t), u_3(t))$ of degree n are mapped via (9) to polynomials of degree $2n$. Taking account of $x_i(t) = \int \dot{x}_i(t)\ dt + C$ with $i = 1, \ldots, 4$, we obtain a polynomial MPH curve of degree $2n+1$. Due to space limitations, we postpone a more detailed description. However, the basic principles of the analytic construction are the same as for MPH curves of $\mathbb{R}^{2,1}$, which are inspected by Moon [9] in detail.

An alternative approach is to construct MPH curves as Laguerre images of PH curves. For example, one could make use of well-known properties on spatial PH cubics [5,6,14] for building MPH cubics. Consider a cubic MPH Bézier curve $\boldsymbol{x}(t)$ with control points $\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3 \in \mathbb{R}^{3,1}$. $\boldsymbol{b}_0$ and $\boldsymbol{b}_1$ resp. $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$ define two hyperbolic tangent vectors $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$, which span a three-dimensional plane Π^3. If there exits a Laguerre transformation τ with $\tau(\Pi^3) = \mathbb{R}^3$, then $\boldsymbol{x}(t) = \tau^{-1}(\boldsymbol{y}(t))$ is a Laguerre image of a (spatial) PH cubic $\boldsymbol{y}(t) \in \mathbb{R}^3$ (that fact comes from properties 2) and 3) of proposition 4 in combination with the convex hull property of Bézier curves).

Proposition 5. *Consider a hyperplane* $\Pi^3 \subset \mathbb{R}^{3,1}$ *spanned by two skew hyperbolic lines* t_1 *and* t_2. *There exists a Laguerre transformation* τ *with* $\tau(\Pi^3) = \mathbb{R}^3$ *iff all lines joining* t_1 *and* t_2 *are hyperbolic.*

Proof: All straight lines in $\mathbb{R}^3$ are hyperbolic. Because of Corollary 2, it is necessary for the existence of τ, that any line of the linear congruence $t_1 \star t_2$ is hyperbolic, too. The sufficiency is proved by construction: With a pe rotation τ_1 one hyperbolic line t_1 can be mapped to $\mathbb{R}^3$. $\tau_1(t_1)$ and the point $\boldsymbol{p} := \tau_1(t_2) \cap \mathbb{R}^3$ span a 2-dimensional plane $\Pi^2 \subset \mathbb{R}^3$. An appropriate Euclidean motion τ_2 maps Π^2 onto the x_2, x_3-plane. Let $\boldsymbol{q} = (q_1, 0, 0, q_4)$ be the intersection point of $\tau_2 \circ \tau_1(t_2)$ with the x_1, x_4-plane. Case 1) $|q_1| < |q_4|$: the pe rotation (5) with $\alpha = \operatorname{arctanh}(-q_1/q_4)$ maps $\tau_2 \circ \tau_1(\Pi^3)$ onto the hyperplane $x_1 = 0$. Since it contains elliptic lines, this contradicts the assumption that $t_1 \star t_2$ is hyperbolic. Case 2) $|q_1| = |q_4|$: $\tau_2 \circ \tau_1(\Pi^3)$ is the hyperplane $x_1 = x_4$. Since it contains parabolic lines, we also get a contradiction. It remains to consider Case 3) $|q_1| > |q_4|$: the pe rotation (5) with $\alpha = \operatorname{arctanh}(-q_4/q_1)$ maps $\tau_2 \circ \tau_1(\Pi^3)$ onto $\mathbb{R}^3$. □

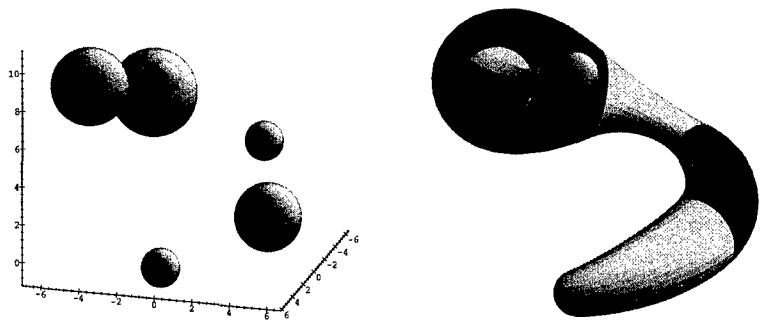


Fig. 6. Interpolation of spheres with cyclide spline.

§6. Interpolation with Canal Surfaces

Finally we briefly discuss simple interpolation problems concerning canal surfaces. The construction of canal surfaces which interpolate given spheres or tangent cones is reduced in the cyclographic model to finding an interpolating curve in $\mathbb{R}^{3,1}$. Therefore, standard algorithms can be generalized from $\mathbb{R}^3$ to $\mathbb{R}^{3,1}$ in order to solve the problem. Indeed, one has to take into account that these methods could generate curves with elliptic tangent vectors, which does not comply with real canal surfaces! This problem can be avoided using algorithms which deal with MPH curves.

Example 6. $n+1$ spheres P_i $(i = 0, \ldots, n)$ can be interpolated with a G^1-continuous cyclide spline (Fig. 6). Each piece is characterized by a pe circle segment with control points $\boldsymbol{b}_{k,i} \in \mathbb{R}^{3,1}$ and weights $\beta_{k,i} \in \mathbb{R}$, $k = 0, 1, 2$. Per given tangent vector $\boldsymbol{t}_{i-1}$ at $\boldsymbol{p}_{i-1}$ the segment is determined uniquely. Thus with any starting vector $\boldsymbol{t}_0$, the pe circular spline curve can be produced successively. To perform the computation, one can generalize an ordinary circular spline algorithm. The formulas (10) are cited from [13], just replacing the Euclidean metric by the pe metric:

$$\begin{aligned} \boldsymbol{b}_{0,i} &= \boldsymbol{p}_{i-1}, \quad \boldsymbol{b}_{1,i} = \boldsymbol{p}_{i-1} + \frac{\|\Delta \boldsymbol{p}_i\|_{pe}^2}{2\langle \Delta \boldsymbol{p}_i, \boldsymbol{t}_{i-1}\rangle_{pe}}, \quad \boldsymbol{b}_{2,i} = \boldsymbol{p}_i, \\ \beta_{0,i} &= 1, \quad \beta_{1,i} = \frac{\langle \Delta \boldsymbol{p}_i, \boldsymbol{t}_{i-1}\rangle_{pe}}{\|\Delta \boldsymbol{p}_i\|_{pe}\|\boldsymbol{t}_{i-1}\|_{pe}}, \quad \beta_{2,i} = 1. \end{aligned} \tag{10}$$

The only condition on the input data is $\|\Delta \boldsymbol{p}_i\|_{pe} := \|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}\| > 0$, i.e., the spheres P_{i-1} and P_i cannot lie inside each other.

Example 7. A piecewise Hermite interpolant can be realized with a cubic MPH canal surface. Consider two spheres P_{i-1}, P_i (specified by points $\boldsymbol{p}_{i-1}$, $\boldsymbol{p}_i$ in $\mathbb{R}^{3,1}$) and tangent cones T_{i-1}, T_i (described as hyperbolic lines $\boldsymbol{t}_{i-1}$,$\boldsymbol{t}_i \in \mathbb{R}^{3,1}$ spanning a hyperplane $\Pi^3 \subset \mathbb{R}^{3,1}$). If this input data satisfy the condition of Proposition 5, then there exists a Laguerre transformation

τ which maps Π^3 to $\mathbb{R}^3$, and a well-known Hermite interpolation algorithm [6,14] produces a cubic PH Hermite interpolant $\boldsymbol{c}(t) \in \mathbb{R}^3$. Its Laguerre image $\tau^{-1}(\boldsymbol{c}(t))$ is a cubic MPH curve and interpolates the original input data.

References

1. Boehm, W., Circles of curvature for curves in space, Comput. Aided Geom. Design **16** (1999), 633–638.
2. Cecil, T. E., *Lie Sphere Geometry*, Springer, New York, 1992.
3. Elber, G., and M. S. Kim, Computing rational bisectors, IEEE Computer Graphics & Applications **19** (1999), 76–81.
4. Farouki, R. T., and T. Sakkalis, Pythagorean Hodographs, IBM J. Res. Dev. **34** (1990), 736–752.
5. Farouki, R. T., and T. Sakkalis, Pythagorean-hodograph space curves, Advances in Computational Mathematics **2** (1994), 41–66.
6. Jüttler, B., and C. Mäurer, Cubic Pythagorean hodograph spline curves and applications to sweep surface modeling, Computer-Aided Design **31** (1999), 73–83.
7. Krasauskas, R., and C. Mäurer, Studying cyclides with Laguerre geometry, Comput. Aided Geom. Design, to appear.
8. Mäurer, C., and R. Krasauskas, Joining cyclide patches along quartic boundary curves, in *Mathematical Methods for Curves and Surfaces II*, M. Dæhlen, T. Lyche, and L. L. Schumaker (eds.), Vanderbilt University Press, Nashville, 1998, 359–366.
9. Moon, H. P., Minkowski Pythagorean hodographs, Comput. Aided Geom. Design **16** (1999), 739–753.
10. Paluszny, M., and W. Boehm, General cyclides, Comput. Aided Geom. Design **15** (1998), 699–710.
11. Peternell, M., and H. Pottmann, Computing rational parametrizations of canal surfaces, Journal of Symbolic Comutation **23** (1997), 255–266.
12. Pottmann, H., and M. Peternell, Applications of Laguerre geometry in CAGD, Comput. Aided Geom. Design **15** (1998), 165–186.
13. Seemann, G., Kreissplines und Schraubliniensplines: Interpolation und Approximation durch Raumkurven mit konstanten Invarianten, PhD-thesis, TU Darmstadt, 1995.
14. Wagner, M., and B. Ravani, Curves with rational Frenet frame motion, Comput. Aided Geom. Design **15** (1997), 79–101.

Christoph Mäurer
TWT-GmbH
Finkenstr. 16
70794 Filderstadt, Germany
fa-twt.christoph.maeurer@daimlerchrysler.com

Geometric Computing with CGAL and LEDA

Kurt Mehlhorn and Stefan Schirra

Abstract. LEDA and CGAL are platforms for combinatorial and geometric computing. We discuss the use of LEDA and CGAL for geometric computing and show that they provide a unique framework for exact, efficient and convenient geometric computing.

§1. Introduction

LEDA (Library of Efficient Data Structures and Algorithms) [16,17] and CGAL (Computational Geometry Algorithms Library) [8,26] are platforms for combinatorial and geometric computing developed in the ESPRIT-projects ALCOM II, ALCOM-IT, CGAL, and GALIA. Concerning geometric computing, the systems provide number types, geometry kernels, geometric algorithms, and visualization. They by now provide a significant fraction of the algorithms and data structures described in the computational geometry literature, where in this context computational geometry subsumes the field covered by the annual ACM Symposia on Computational Geometry. The systems are designed such that it is easy to build programs on top of them. The computations in LEDA and CGAL are exact, i.e., behave according to their mathematical specifications. This is a strong point of both systems, distinguishing them form many other geometric software products.

Based on the insight that algorithm design must include implementation to have maximal impact, Kurt Mehlhorn and Stefan Näher started the development of the LEDA software library of efficient data structures and algorithms in Saarbrücken in '89 using C++ as programming language. LEDA is now developed at Max-Planck-Institut für Informatik, Saarbrücken (Germany), and Martin-Luther-Universität Halle-Wittenberg (Germany). The idea of CGAL was conceived in fall of '94, inspired by the success of LEDA and in order to bundle forces previously put into predecessors of CGAL [2,11,20]. Development of CGAL was started in fall '96 in the CGAL-project and is

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 277–286.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

now continued in the GALIA-project. GALIA is carried out by Max-Planck-Institut für Informatik, Saarbrücken, ETH Zürich (Switzerland), Freie Universität Berlin (Germany), INRIA Sophia-Antipolis (France), Martin-Luther-Universität Halle-Wittenberg, Tel-Aviv University (Israel), and Utrecht University (The Netherlands). The goal is to make the most important of the solutions and methods developed in computational geometry available to users in industry and academia in a C++ library.

§2. The Need for a Geometry Software Library

Reusing code that already exists and is used and thereby tested rather than implementing everything from scratch saves development time and hence reduces cost [5]. It also eases maintenance of code. Software libraries also ease the transfer of state-of-the-art algorithmic knowledge into application areas. Since geometric computing is a wide area, many application areas can benefit from the availability of the re-usable code of a geometry software library. The importance of libraries of software components in subject area domains is clearly stated in a recent report of the information technology advisory committee of the president of the US [22].

In geometric computing, software libraries consisting of reliable components are particularly useful, since implementors of geometric algorithms are faced with notoriously difficult problems [18], especially the problems of robustness and degeneracies.

Robustness

Theory usually assumes exact computation with arbitrary real numbers, while the standard substitution for real numbers in scientific computing in practice, floating-point arithmetic, is inherently imprecise. In practice, implementations of geometric algorithms compute garbage or completely fail more or less occasionally, because rounding errors lead to wrong and contradictory decisions, see [14,25,27]. With floating-point arithmetic, basic laws of arithmetic, on which the correctness proof of geometric algorithms is based, of course, don't hold anymore. We invite the reader to carry out the following simple experiment: Compute the point of intersection of the two lines with built-in floating point arithmetic. Then, again using built-in floating point arithmetic, check whether the computed intersection point lies on the intersecting lines.

Figure 1 shows an incorrect result of a computation due to rounding errors. The task is to compute the extreme points of intersection points of a set of line segments, where a point is called extreme with respect to a set of points if its removal from the set changes the convex hull of the point set. The line segments have randomly chosen endpoints lying on a circle. In a first step the intersection points of the line segments are computed, then a convex hull algorithm is run on the points computed in the first step. With floating-point arithmetic, some collinearities are not detected and too many extreme points are reported. Extreme points are shown as small disks in Figure 1. The points surrounded by a circle are actually not extreme. In the problem considered

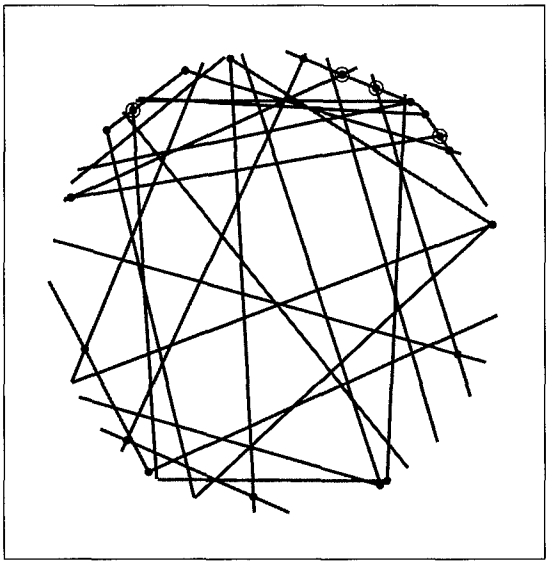


Fig. 1. Extreme points among intersection points of 30 line segments.

here, the computed output might still be useful; for many other geometric problems, however, failures of pure floating-point based implementations are much more drastically. They just crash.

Adding epsilons by trial and error to equality tests used to be common practice in implementations of geometric algorithms, but it in no way leads to a reliable correct implementations. Two main approaches to solving the precision-caused robustness problem can be identified. The first is re-designing algorithms such that they can deal with imprecision, e.g. compute a good approximate solution, but never crash. So far, this approach has been applied successfully to only very few basic problems in computational geometry, see [14,25,27]. The second approach is exact geometric computation [28], which means computing with such a precision that an implementation behaves like its theoretical counterpart, and therefore according to its mathematical specification. This is possible for many geometric problems, at least, theoretically. Note that in practice, the input does not involve arbitrary real numbers. Of course, exact geometric computation slows down computation, but thanks to clever adaptive computation using floating-point arithmetic whenever known to produce the correct result [10,15], it is now much closer to the speed of floating-point computation than it used to be a decade ago. Since libraries must be reliable in order to be usable in general, the exact geometric computation approach is taken in LEDA and in CGAL.

Degeneracies

Robustness problems caused by rounding errors are closely related to degeneracies, i.e. "exceptional" input configurations. Theory often neglects degen-

eracies for the sake of a cleaner exposition of an algorithm, and also because they are rare from a theoretical point of view: they have measure zero in the set of all possible inputs over the real numbers. In practice, however, they occur frequently. Since theory papers often leave handling degenerate cases as an exercise to the reader, implementors are often left alone with the burden of investigating the details of handling degeneracies. Furthermore, this leads to treating degeneracies as an afterthought, which is, according to our experience [3], not the most suitable way to think about them, since it leads to unnecessarily complicated and blown-up code. Considering degeneracies right from the beginning seems to be a much better approach.

Symbolic perturbation schemes have been proposed as a general approach to removing degeneracies, for an overview see [24]. With this approach, the input is perturbed symbolically such that no degeneracies arise anymore. The perturbed input can then be processed by an algorithm assuming general position. The computed output, however, does not correspond to the actual input, but to the perturbed input. Therefore, the complexity of the output might be much larger than the output for the actual input [3]. For some problems, the symbolic perturbation approach works out fine; for others, a postprocessing step is required to deduce the actual output from the output computed for the perturbed input. In many cases, this is a non-trivial task, as hard as dealing with degeneracies directly.

Algorithms and data structures in CGAL and LEDA handle all possible degenerate case by default. So a user need not to worry about all the degenerate cases. If an algorithm or data structure should not handle a degenerate case, this is clearly stated in the documentation and this precondition is checked in the implementation. However, mainly to support rapid prototyping, CGAL also provides tools for symbolic perturbation. A general randomized symbolic perturbation scheme is available for the CGAL kernels [6]. A new promising approach that has been started within the CGAL project, is controlled perturbation [23]. Here the input is perturbed numerically, such that general position is guaranteed.

§3. Number Types

The lowest level in geometric computing is the arithmetic level. LEDA and CGAL provide various number types to support exact geometric computation. LEDA provides `leda_integer`, a number type for arbitrary precision integer arithmetic and `leda_rational`, a number type for arbitrary precision rational arithmetic, based on `leda_integer`. Furthermore, it provides `leda_bigfloat`, a number type for floating point arithmetic with extended precision. A user can choose the mantissa length of the `leda_bigfloats` or let the number type increase the mantissa length on demand. The most sophisticated number type in LEDA is the type `leda_real` [4]. This number types models a subset of algebraic numbers: All integers are `leda_reals` and `leda_reals` are closed under the operations $+$, $-$, $\cdot$, $/$, and $\sqrt[k]{}$. `leda_reals` record the computation history in an expression dag, and use adaptive evaluation to guarantee

that all comparison operations give the correct results. They use `bigfloats` internally. LEDA and CGAL also provide interval arithmetic. Furthermore, CGAL provides some fixed point arithmetic based on built-in `floats`, as well as wrappers for the gnu multiple precision integer arithmetic [12] and the number types provided by CLN [13].

§4. Geometry Kernels

The kernel of a geometry library contains the basic geometric objects and basic operations on them like points, lines, planes, sideness test, intersection and distance computations.

LEDA provides an exact geometry kernel for rational computations. Internally, it uses floating-point filters [10,15] to speed up exact computation. With floating-point filters, an expression whose sign has to be computed is first evaluated using floating-point arithmetic. Moreover, an upper bound on the error of the floating-point computation is computed as well. By comparison of the absolute computed floating-point value with this error bound, it is checked whether the floating-point computation is guaranteed to be reliable. If the sign cannot be deduced with floating-point arithmetic, the expression is re-evaluated with a more reliable arithmetic. In case of the rational geometry kernel in LEDA, arbitrary precision integer arithmetic is used. The rational geometry kernel of LEDA uses homogeneous coordinates and is coupled to the number types `leda_rational` and `leda_integer`.

CGAL provides two families of geometry kernels, one based on Cartesian coordinates and one based on homogeneous coordinates [9]. Both kernels are parameterized by a number type. All number types fulfilling a very small list of requirements can be used with the CGAL kernels. For example, the user might choose the Cartesian kernel with rational arithmetic or the homogeneous kernel with integer arithmetic. In particular, for computations involving k-th root operations, the number type `leda_real` can be used with the CGAL kernels. There are also number types that use floating-point filter techniques using interval arithmetic to speed up exact computation. These number types assume that the data passed to a test function are exact. Hence, this technique is not suited for cascaded computations. The parameterization allows a user to choose the arithmetic according to the actual needs. Using a CGAL kernel with `leda_real` is certainly the most convenient way to get reliable computation.

LEDA also provides a kernel that uses double precision floating-point arithmetic internally. Similarly, the CGAL kernels can be used with built-in floating point number types as well. This might be sufficient for some problems, but since correctness can not be guaranteed, the use of these kernels is not recommended in general.

§5. Geometric Algorithms and Data Structures

CGAL and LEDA by now provide a significant fraction of the algorithms and data structures described in the computational geometry literature. They pro-

vide several algorithms to compute convex hulls in low and higher dimensions. They provide algorithms and data structures for triangulations, constrained triangulations, Delaunay triangulations, regular triangulations, and Voronoi diagrams in two-dimensional space and Delaunay triangulations and regular triangulations in three-dimensional space. Furthermore they provide several algorithms for line segment intersection and regularizing Boolean operations on polygons. CGAL and LEDA also provide a number of query data structures. For example, there are range- and segment trees, kd-trees, as well as a data structure for range and nearest neighbor queries based on Delaunay triangulations. CGAL provides data structures for polyhedral surfaces based on a half-edge data structure, it provides a topological map data type and a planar map data structure, and a data structure for arrangements. The libraries also contain algorithms for curve reconstruction in the plane.

CGAL and LEDA provide algorithms for a number of geometric optimization problems. There are algorithms computing smallest enclosing circles, smallest enclosing ellipses, and smallest enclosing spheres, the latter in any dimension. Furthermore, there are a number of algorithms based on matrix search like computing extremal polygons and rectangular p-centers. LEDA also contains algorithms for computing smallest enclosing annuli with respect to area and width, and algorithms to compute minimum spanning trees for a set of points.

The geometric algorithms of LEDA come in two versions, one using the exact rational geometry kernel and one using the unreliable floating-point kernel. CGAL's algorithms and data structures are even more flexible with respect to the geometry kernel used. All algorithms and data structures of CGAL are parameterized by a template parameter called traits class. This traits class provides an algorithm or data structure with all the type information it needs. It tells the algorithm on which types it should operate and which types it should use to do that.

The parameterization and the resulting genericity of CGAL's algorithms and data structures is best illustrated by a simple, but instructive example. Computing the convex hull of a set of points in the plane is an intensively studied problems in computational geometry. The input is a set of points in the plane, the output is the counterclockwise sequence of extreme points. Andrew's variant [1] of the Graham scan algorithm can be formulated in such a way that it needs only two primitive operations on the points, namely a primitive to compare two points in order to sort the points lexicographically by their Cartesian coordinates, and a primitive to check the order type of a triple of points, more precisely, to check whether a sequence of three points forms a left turn. The CGAL implementation of Andrew's algorithm is parameterized by a point type and the two required primitive operations. The latter two are passed as function object types and need to correspond to the point type. We call the parameter types of an algorithm or data structure the traits types. To avoid long lists of template parameters, the traits types are collected in the traits class. Note that the parameterization is on the level of data types, not on the level of objects. In order to use the CGAL implementation of

Andrew's algorithm with a point type from a CGAL kernel, no traits class needs to be specified. CGAL adds an appropriate one. If a user wants to run the algorithm on a different point type, for example, a point type from some other C++ library or system, for example from LEDA or from some Geographical Information System, an appropriate traits class for this point type must be passed to the function in order to tell it which operations it should use. That's it. Given such a traits class, the algorithm now works with non-CGAL types as well. CGAL provides traits classes for both LEDA kernels.

The parameterization by a traits class can be used to avoid explicit transformation of the data. Assume that we have points in three dimensional space. Using a traits class that provides a comparison primitive and an order type predicate that both operate on x and y coordinates of the points only, the CGAL implementation of Andrew's algorithm can be used to compute the sequence of three-dimensional points whose projections onto the xy-plane form the convex hull of the projections of all points onto that plane. There is no need to explicitly transform the points into two-dimensional points. With an appropriate traits class, the algorithm can directly operate on the three-dimensional points. This saves time and space.

This feature is most likely even more interesting for Delaunay triangulations. Assume we have a triangulated irregular network (TIN), and we want to make a TIN with the same set of vertices without long and skinny triangles. This is usually accomplished by computing the two-dimensional Delaunay triangulation of the projections of the vertices of the TIN and lifting the vertices and triangles again. CGAL allows you to do this without explicit projection using an appropriate traits class. There are further examples where traits classes can be used nicely in the context of geometric transformations.

§6. Visualization

In LEDA, there is a data type `leda_window` which provides an interface for graphical input and output of basic geometric objects for both the X11 system on Unix platforms and Microsoft's Windows systems. This data type works with the basic geometric objects of both CGAL and LEDA. CGAL also provides preliminary support for graphical output via `OpenGL` and `geomview`.

A recent addition to CGAL and LEDA is the data type `GeoWin`. It provides an interface for the visualization of the result and progression of geometric algorithms using the window data type of LEDA. A `GeoWin` is an editor for sets of geometric objects. GeoWin manages the geometric objects in so called scenes. A scene contains a container storing geometric objects (the contents of the scene) and provides all operations that GeoWin needs to edit it. A `geo_scene` maintains a container with geometric objects. The `GeoWin` data type can be used for the construction and display of geometric objects and data structures, the visualization of geometric algorithms, writing interactive demos for geometric algorithms and debugging geometric algorithms.

§7. Conclusions

Reliability means that software behaves as specified. Unfortunately, there are many exceptions to this rule for geometric software, mainly due to the issues discussed in Section 2. Correctness and reliability are even more important for the components of a software library. You might be willing to accept shortcomings of a program designed for a special purpose, if problematic input instances never arise in your context. Since library component need to be generally applicable, any such shortcomings are not acceptable. CGAL (if used with a number type for exact geometric computation) and LEDA (with the rational kernel) provide geometric software that behaves according to its mathematical specification. That makes it easy to combine components form these libraries, and to build larger entities out of these components.

The use of exact computation alone cannot guarantee correctness. CGAL and LEDA also use program checking [19] to increase reliability of its components. A program checker need not compute the output for a given input. It already gets both input and output, and then has to verify that the output is the correct output for the given input. While a program gets x and has to compute $f(x)$, a checker gets x and y and must only check whether $y = f(x)$. The latter step should be computationally simpler, such that it is less likely that its implementation is buggy.

At present, LEDA and CGAL consists of more than 100,000 lines of C++ code each. Neither library provides class libraries in the sense of Smalltalk, but both provide fairly small class hierarchies if any. CGAL uses the generic programming paradigm that became known with the Standard Template Library (STL), which is now part of Standard C++. This makes CGAL very flexible, more flexible than LEDA. On the other hand, LEDA is a more complete, closed programming framework that also contains very useful components for combinatorial computing. Due to its generic design, CGAL is more open. It often relies on other sources for basic non-geometric data structures, mainly on the STL. Due to its generic design, it works well together with LEDA. Since CGAL has a more modern design and is developed by a larger group of people, the future will certainly be with CGAL. However, LEDA's components for geometric computing will continue to be useful, especially within CGAL. For more information and to download LEDA, see

`http://www.mpi-sb.mpg.de/LEDA`

For more information and to download CGAL, see

`http://www.cs.uu.nl/CGAL`

Acknowledgments. Work on this paper has been supported by ESPRIT-IV LTR project 28155 (GALIA).

References

1. Andrew, A. M., Another efficient algorithm for convex hulls in two dimensions, Inform. Process. Lett. **9** (1979), 216–219.
2. Avnaim, F., C++GAL: A C++ Library for geometric algorithms, INRIA Sophia-Antipolis, 1994.
3. Burnikel, C., K. Mehlhorn, and S. Schirra, On degeneracy in geometric computations, Proc. of the 5th ACM-SIAM Symp. on Discrete Algorithms, 1994, 16–23.
4. Burnikel, C., R. Fleischer, K. Mehlhorn, and S. Schirra, Efficient exact geometric computation made easy, in Proc. 15th Annu. ACM Sympos. Comput. Geom., 1999, 341–350.
5. Carroll, M. D., M. A. Ellis, *Designing and Coding Reusable C++*, Addison-Wesley, 1995.
6. Comes, J., and M. Ziegelmann, An easy to use implementation of linear perturbations within CGAL, in Algorithm Engineering, WAE99, Lect. Notes in Comp. Science vol. 1668, Springer Verlag, 1999, 169–182.
7. Epstein, P., J. Kavanagh, A. Knight, J. May, T. Nguyen, and J.-R. Sack, A workbench for computational geometry, Algorithmica **11** (1994), 404–428.
8. Fabri, A., G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr, On the design of CGAL, a computational geometry algorithms library. Software – Practice & Experience, special issue on Algorithm Engineering
9. Fabri, A., G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schönherr, The CGAL Kernel: A basis for geometric computation, in Workshop on Applied Computational Geometry (WACG'96), Lect. Notes in Comp. Science, vol. 1148, 1996, 191–202.
10. Fortune, S., and C. van Wyk. Static analysis yields efficient exact integer arithmetic for computational geometry, ACM Transactions on Graphics, **15** (1996), 223–248.
11. Giezeman, G. J., PlaGeo, a library for planar geometry, and SpaGeo, a library for spatial geometry, Utrecht University, 1994.
12. Granlund, T., GNU MP, The GNU multiple precision arithmetic library, 2.0.2 edition, June 1996.
13. Haible, B., CLN, The class library for numbers, 1.0.1 edition, June 1999. `http://clisp.cons.org/~haible/packages-cln.html`.
14. Hoffmann, C., The problem of accuracy and robustness in geometric computation, IEEE Computer, March 1989, 31–41.
15. Karasick, M., D. Lieber, and L. Nackman, Efficient Delaunay triangulation using rational arithmetic, ACM Transactions on Graphics, **10** (1991), 71–91.
16. Mehlhorn, K., S. Näher, M. Seel, and C. Uhrig, The LEDA user manual, version 4.0, 1999.

17. Mehlhorn, K., and S. Näher. *The LEDA Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.
18. Mehlhorn, K., and S. Näher, The implementation of geometric algorithms, 13th World Computer Congress IFIP94, volume 1. Elsevier Science B.V. North-Holland, Amsterdam, 1994, 223–231.
19. Mehlhorn, K., S. Näher, T. Schilz, S. Schirra, M. Seel, R. Seidel, and C. Uhrig, Checking Geometric Programs or Verification of Geometric Structures Computational Geometry: Theory and Applications **12** (1999), 85-103.
20. Nievergelt, J., P. Schorn, M. de Lorenzi, C. Ammann, and A. Brüngger, XYZ: A project in experimental geometric computation, Computational Geometry: Methods, Algorithms and Applications, Lect. Notes in Comp. Science vol. 553. Springer-Verlag, 1991, 171–186.
21. Overmars, M., Designing the computational geometry algorithms library CGAL, in *Applied Computational Geometry : Towards Geometric Engineering (WACG96)*, M. C. Lin and D. Manocha (eds.), Lect. Notes in Comp. Science vol. 1148, 1996, 53–58.
22. Presidents Information Technology Advisory Committee, Interim Report to the President, sect. 3.1.2, `http://www.ccic.gov/ac/interim/`, 1998.
23. Raab, S., Controlled perturbation for arrangements of polyhedral surfaces with application to swept volumes, in Proc. 15th ACM symposium on Computational Geometry, 1999, 163–172.
24. Seidel, R., The nature and meaning of perturbations in geometric computing, Proc. 11th Sympos. Theoret. Aspects Comput. Sci., Lect. Notes Comp. Science vol. 775, Springer Verlag, 1994.
25. Schirra, S., Precision and robustness issues in geometric computation, Handbook on Computational Geometry, Elsevier Science Publishers, Amsterdam, The Netherlands, 1999.
26. Schirra, S., R. Veltkamp, M. Yvinec (Eds.), CGAL reference manuals, version 2.1, 1999.
27. Yap, C. K., Robust geometric computation, in *CRC Handbook on Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke (eds.), CRC Press, 1997, 653–668.
28. Yap, C. K., and T. Dubé, The exact computation paradigm, in *Computing in Euclidean Geometry*, 2nd edition, D. Du and F. Hwang (eds.), World Scientific Press, 1995, 452–492.

Kurt Mehlhorn and Stefan Schirra
Max-Planck-Institut für Informatik
66123 Saarbrücken
Germany
`mehlhorn@mpi-sb.mpg.de`
`stschirr@mpi-sb.mpg.de`

A Geometric Approach for Knot Selection in Convexity-Preserving Spline Approximation

R. Morandi, D. Scaramelli, and A. Sestini

Abstract. A geometric approach is proposed for selecting the knots used in a parametric convexity-preserving B-spline approximation scheme. The approach automatically gives the necessary information about the shape suggested by the data which may be exact or not.

§1. Introduction

In many different fields such as medicine, physics, engineering and computer graphics the amount of data obtained through experimental and/or statistical surveys is very large. Consequently, the selection of a suitable small set of knots becomes an indispensable step within any efficient spline approximation scheme.

In this paper, large sets of exact and non-exact data points are approximated by means of a spline approximation scheme. So a knot selection strategy is necessary and this is provided by a geometric approach. In detail, the proposed approach is based on some weights suitably associated to the data points and directly computed from them. In addition, the method automatically defines the shape suggested by the data, here assumed planar and exact or not. Furthermore, the shape constraints for the approximating curve can be obtained in order to reproduce the desired behaviour.

Several other approaches have been studied, such as the knot removal methods [5,6], to reduce the number of parameters involved in an approximation problem. Interesting results are already available even for constrained approximation [1]. However, the approach introduced here differs from those because it does not reduce an initial large set of knots, but directly computes a suitable set. Furthermore, it does not require the starting approximation with many knots used in [1,5,6] for the weight definition (solving a minimization problem for each weight).

The proposed strategy has been tested on several examples for open and closed curves, and for exact and non-exact data points. The approximating

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 287–296.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

curve is obtained by means of a convexity-preserving B-spline approximation scheme. The goodness of fit of the approximation has been estimated measuring the mean square distance of the data points from the resulting curve.

The outline of the paper is as follows. The problem and the method are presented in the next section. The strategies used to define the shape suggested by the data and to select the knots are introduced in Section 3, and they are given in more detail in the Appendix. Finally, in Section 4 the numerical results obtained for four sets of data points are given.

§2. The Problem and the Method

Let $\boldsymbol{P}_i \in \mathbb{R}^2, i = 0, \ldots, n$, be exact or non-exact data points, with n very large, and let $\mathcal{T} = \{t_i \in \mathbb{R}, i = 0, \ldots, n\}$ be an assigned set of associated strictly increasing parameter values. For non-exact data, let d be a positive assigned quantity so that $||\boldsymbol{P}_i - \boldsymbol{P}_i^e||_2 \leq d, i = 0, \ldots, n$, where $\boldsymbol{P}_i^e$ is the (unknown) exact data point corresponding to $\boldsymbol{P}_i$. We observe that, for simplicity's sake, the same maximum error value is assumed for all the data points.

It is well known that, if the data sets are very large and in particular the data are non-exact, the use of an approximation scheme is the only reasonable approach to construct a curve with the desired behaviour. Thus, here the aim is to first define the convexity constraints suggested by the data, and then to give a strategy for selecting a suitable small number of knots to construct a convexity-preserving least-square B-spline approximating curve.

Let $N_{jk}(t), j = 1-k, \ldots, nr-1$, be the usual B-splines of order k [4] defined with an extended knot vector $\Theta^* = \{\tau_{1-k}, \ldots, \tau_0, \ldots, \tau_{nr}, \ldots, \tau_{nr+k-1}\}$. Thus, we can introduce the B-spline representation of a spline curve

$$\boldsymbol{C}(t) = \sum_{j=1-k}^{nr-1} \boldsymbol{Q}_j N_{jk}(t), \qquad t \in [\tau_0, \tau_{nr}], \tag{1}$$

where $\boldsymbol{Q}_j, j = 1-k, \ldots, nr-1$ are de Boor control points.

The problem can be divided into three sub-problems. The definition of the shape suggested by the data (that is the determination of the convexity changes required to the approximating curve), the selection of the knots, and finally the construction of the convexity-preserving least-square B-spline approximating curve.

In particular, the shape suggested by the data is obtained through the procedure called "Shape Determination" (SD) described in the Appendix. SD uses some coefficients $u_i, i = 0, \ldots, n$, suitably associated with the data to establish in which parameter values zero curvature is required, and to determine the curvature sign in the interval $[\tau_0, \tau_{nr}]$. As the planar case is here considered, the curvature is defined as the function $\rho(t) = \frac{\dot{\boldsymbol{C}}(t) \times \ddot{\boldsymbol{C}}(t)}{||\dot{\boldsymbol{C}}(t)||_2^3}$, where $\boldsymbol{v} \times \boldsymbol{z} = v_1 \cdot z_2 - v_2 \cdot z_1, \ \forall \boldsymbol{v}, \boldsymbol{z} \in \mathbb{R}^2$). Thus, the procedure "Knots Selection" (KS) described in the Appendix selects the knot vector $\Theta = \{\tau_0, \ldots, \tau_{nr}\}$ by using the weights $w_i = |u_i|, i = 0, \ldots, n$. Θ^* is defined as the corresponding

extended knot vector, taking into account whether the approximating curve is open or closed.

Finally, the last step is realized through the solution of a constrained parametric least-squares problem as a general constrained optimization problem. In fact, as the general parametric case is considered, the objective function $\sum_{i=0}^{n} ||\boldsymbol{P}_i - \boldsymbol{C}(t_i)||_2^2 = \sum_{i=0}^{n} ||\boldsymbol{P}_i - \sum_{j=1-k}^{nr-1} \boldsymbol{Q}_j N_{jk}(t_i)||_2^2$ is quadratic in the unknowns $\boldsymbol{Q}_j, j = 1-k, \ldots, nr-1$, but the convexity constraints are nonlinear.

§3. Data Shape Determination and Knot Selection Strategy

The SD procedure defines the shape suggested by the data, and KS selects the knots for constructing the B-spline curve. They are presented in the Appendix, but are commented upon here. Concerning the shape determination, SD computes the set $\mathcal{U} = \{u_0, \ldots, u_n\}$ whose sign variations are the curvature sign variations required for the approximating curve. If exact data are considered, $|u_i|$ is the reciprocal of the radius of the circle which passes through the points $\boldsymbol{P}_{i-1}, \boldsymbol{P}_i, \boldsymbol{P}_{i+1}$, and its sign is that of $\Delta_i = \frac{1}{2} det(\boldsymbol{L}_i, \boldsymbol{L}_{i+1})$, where $\boldsymbol{L}_i = \boldsymbol{P}_i - \boldsymbol{P}_{i-1}$. On the other hand, the sign of Δ_i is a reliable geometric information in the case of non-exact data only if the condition (2) of the theorem given in this section holds. Thus, the definition of $\mathcal{U}$ is suitably modified for non-exact data, using an input tolerance tol_d and considering the result given in the theorem below. In this case u_i is defined using the circle through the points $\boldsymbol{P}_{l_i}, \boldsymbol{P}_i$ and $\boldsymbol{P}_{r_i}$, where $\boldsymbol{P}_{l_i}$ and $\boldsymbol{P}_{r_i}$ are suitably selected points. In detail, $l_i \leq i-1$, $r_i \geq i+1, \sum_{k=l_i}^{i-1} ||\boldsymbol{P}_{k+1} - \boldsymbol{P}_k||_2 < tol_d \cdot Lp$ and $\sum_{k=i}^{r_i-1} ||\boldsymbol{P}_{k+1} - \boldsymbol{P}_k||_2 < tol_d \cdot Lp$, where Lp is the length of the polygonal joining the data points. SD computes the set $\Theta_S = \{\tau_0^s, \ldots, \tau_{ns}^s\} \subset \mathcal{T}$ and the set $\Sigma = \{\sigma_0, \ldots, \sigma_{ns-1}\}$, where Θ_S is such that $\tau_0^s = t_0, \tau_{ns}^s = t_n$ and zero curvature is required at each $\tau_i^s, i = 1, \ldots, ns-1$. The desired curvature sign between τ_i^s and τ_{i+1}^s is given by σ_i equal to -1 or 0 or 1. We observe that, for $d \neq 0$, in the procedure $u_i \neq 0$ is assumed to imply the existence of $k_i \in \{i-2, i-1, i\}$ such that $u_{k_i} \cdot u_i > 0, u_{k_i+1} \cdot u_i > 0$ and $u_{k_i+2} \cdot u_i > 0$. This hypothesis seems to be quite reasonable as n is considered very large.

Theorem 1. *Let $\boldsymbol{P}_i \in \mathbb{R}^2$ for $i = 0, \ldots, n$ be assigned non-exact data points, and let d be a small positive assigned quantity such that $||\boldsymbol{P}_i - \boldsymbol{P}_i^e||_2 \leq d, i = 0, \ldots, n$, where $\boldsymbol{P}_i^e$ is the (unknown) exact data point corresponding to $\boldsymbol{P}_i$. If $d < \frac{1}{2} \min_{i=1,\ldots,n} ||\boldsymbol{L}_i||_2$ with $\boldsymbol{L}_i = \boldsymbol{P}_i - \boldsymbol{P}_{i-1}$ and the condition*

$$\frac{||\boldsymbol{N}_i||_2}{||\boldsymbol{L}_i||_2 + ||\boldsymbol{L}_{i+1}||_2} > \frac{11}{4} d, \qquad i = 1, \ldots, n-1, \tag{2}$$

holds, then

$$\boldsymbol{N}_i^e \cdot \boldsymbol{N}_i > 0, \; i = 1, \ldots, n-1,$$

where $\boldsymbol{N}_i = \boldsymbol{L}_i \wedge \boldsymbol{L}_{i+1}$, $\boldsymbol{N}_i^e = \boldsymbol{L}_i^e \wedge \boldsymbol{L}_{i+1}^e$, $\boldsymbol{L}_i^e = \boldsymbol{P}_i^e - \boldsymbol{P}_{i-1}^e$ and the symbols " $\wedge$ " and "$\cdot$" denote the usual vector and scalar product, respectively.

Proof: We can write $\boldsymbol{P}_i^e = \boldsymbol{P}_i + \varepsilon_i \boldsymbol{v}_i$ with $0 \leq \varepsilon_i \leq d$ and $||\boldsymbol{v}_i||_2 = 1$. Then we have $\boldsymbol{L}_i^e = \boldsymbol{L}_i + \varepsilon_i \boldsymbol{v}_i - \varepsilon_{i-1} \boldsymbol{v}_{i-1}$ and $\boldsymbol{N}_i^e = \boldsymbol{N}_i + z_i$, where $z_i = \varepsilon_i \boldsymbol{v}_i \wedge \boldsymbol{L}_{i+1} -$

$\varepsilon_{i-1}\boldsymbol{v}_{i-1} \wedge \boldsymbol{L}_{i+1} + \varepsilon_{i+1}\boldsymbol{L}_i \wedge \boldsymbol{v}_{i+1} + \varepsilon_i\varepsilon_{i+1}\boldsymbol{v}_i \wedge \boldsymbol{v}_{i+1} - \varepsilon_{i-1}\varepsilon_{i+1}\boldsymbol{v}_{i-1} \wedge \boldsymbol{v}_{i+1} - \varepsilon_i\boldsymbol{L}_i \wedge \boldsymbol{v}_i + \varepsilon_{i-1}\varepsilon_i\boldsymbol{v}_{i-1} \wedge \boldsymbol{v}_i$. Thus, we can write $\boldsymbol{N}_i^e = (1 + f(\varepsilon_{i-1}, \varepsilon_i, \varepsilon_{i+1}))\boldsymbol{N}_i$, where $f(\varepsilon_{i-1}, \varepsilon_i, \varepsilon_{i+1}) = \frac{z_i \cdot \boldsymbol{N}_i}{||\boldsymbol{N}_i||^2}$. As a consequence, the assertion holds if $(1 + f(\varepsilon_{i-1}, \varepsilon_i, \varepsilon_{i+1}))||\boldsymbol{N}_i||_2^2 > 0$, that is if $f(\varepsilon_{i-1}, \varepsilon_i, \varepsilon_{i+1}) > -1$. Now with some algebra the following inequality can be easily obtained:

$$f(\varepsilon_{i-1}, \varepsilon_i, \varepsilon_{i+1}) \geq -\frac{d}{||\boldsymbol{N}_i||_2}[2(||\boldsymbol{L}_i||_2 + ||\boldsymbol{L}_{i+1}||_2) + 3d]. \tag{3}$$

So, as $d < \frac{1}{2}\min_{i=1,\ldots,n}||\boldsymbol{L}_i||_2$, $d < \frac{||\boldsymbol{L}_i||_2 + ||\boldsymbol{L}_{i+1}||_2}{4}$. Then, the inequality (3) and the hypothesis (2) imply that $f(\varepsilon_{i-1}, \varepsilon_i, \varepsilon_{i+1}) \geq -\frac{d}{||\boldsymbol{N}_i||_2}(\frac{11}{4}(||\boldsymbol{L}_i||_2 + ||\boldsymbol{L}_{i+1}||_2)) > -1$, thus proving the theorem. □

Concerning the knot selection, in the KS procedure the knot vector $\Theta \subset \mathcal{T}$ is initialized with Θ_S. Then, for each interval $[\tau_{j-1}^s, \tau_j^s], j = 1, \ldots, ns$, it is checked if other parameters must be inserted in the knot vector using weights $w_i = |u_i|, i = 0, \ldots, n$. More precisely, a parameter value $t_l \in \mathcal{T} \cap (\tau_{j-1}^s, \tau_j^s)$ is inserted in Θ if one of the following conditions holds: either the corresponding weight w_l is big enough and $\boldsymbol{P}_l$ is far enough from all the other data related to the parameters previously introduced between τ_{j-1}^s and τ_j^s, or the corresponding weight w_l is not big enough but $\boldsymbol{P}_l$ is too far from them. For choosing reasonable values for the tolerances used in the previous consideration, denoted as tol_w, tol_{d1} and tol_{d2}, it is assumed that the distances are relative to an approximated curve length and the weights are relative to the maximum weight. The parameter values between τ_{j-1}^s and τ_j^s are ordered according to a decreasing order of the corresponding weights.

§4. Numerical Results

Four numerical tests are presented to analyze the performance of the approach. For all the considered tests the parameter values $t_i, i = 0, \ldots, n$, have been computed with the chord-length approach and a scaling such that $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = 1$. The approximated curve length L required as input by the KS procedure has been computed as the length of the piecewise linear interpolant of all the data points if $d = 0$ and of a suitably selected subset of them if $d > 0$.

A sequential quadratic programming method [2] is used to construct the approximating curve by means of the routine CONSTR of the Optimization toolbox of the Matlab package [3]. The set of control points used to start is chosen as the set of data points corresponding to the selected knots. Concerning the constraints, as $k = 4$ has been used in the experiments, four consecutive control points are required to generate a convex polygon if convexity is looked for in the corresponding curve segment. In addition three sequential collinear control points are required if zero curvature is asked at the corresponding knot.

For each test, a figure is given showing the corresponding approximating curve on the left, and the related curvature plot on the right. In all the figures the data points $\boldsymbol{P}_0, ..., \boldsymbol{P}_n$ are denoted with the symbol " · ", the points associated with the knots with the symbol " o ", and the points corresponding to the knots belonging to Θ_S with the symbol " ⊗ ".
The results are summarized in Table 1, where $n+1$ is the number of data points, tol_d is the input tolerance used in the procedure SD, tol_w, tol_{d1} and tol_{d2} are the input tolerances used in the procedure KS and $nr+1$ is the cardinality of the knot vector Θ. The symbol % denotes the percentage ratio $(nr+1)/(n+1)$. To estimate the goodness of fit of the approximation, the Mean Square Distance (MSD) of the data points $\boldsymbol{P}_i, i = 0, \ldots, n$, from the approximating curve is also given in the table.

	Test 1	**Test 2**	**Test 3**	**Test 4**
$n+1$	285	257	126	244
tol_d	-	0.2	0.2	0.5
tol_w	0.11	0.20	0.10	0.50
tol_{d1}	0.015	0.018	0.018	0.037
tol_{d2}	0.05	0.80	0.50	0.90
$nr+1$	36	30	22	8
%	12.6	11.7	17.5	3.3
MSD	1.08e-06	1.80e-03	2.19e-04	1.06e-01

Tab. 1. Results of the tests.

In the first test, data are considered exact ($d = 0$), while in the other tests they are non-exact.

Test 1 relates to a set of 285 exact data points that represent the alphabet capital letter " D ". In this case only a curvature sign variation to the approximating curve is required, as the curvature plot shows.

In Test 2, 257 non-exact data points are considered. They have been obtained by introducing a simulated random perturbation with $d = 0.4$ on the ordinates of the points $(x_i^e, y_i^e), i = 0, \ldots, 256$ defined as $x_i^e = -8 + dx \cdot i$, $y_i^e = 12\frac{sin(R_i)}{R_i}, i = 0, \ldots, 256$, where $R_i = sqrt(2(x_i^e)^2) + eps$, with eps denoting the round-off error and $dx = 0.0625$. We can observe that the simulated error does not preserve the symmetry of the data, and therefore the selected knots are not symmetric.

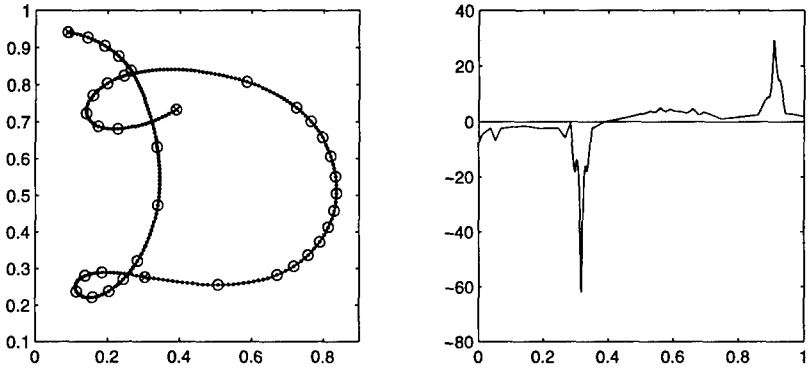


Fig. 1. Test 1: On the left the approximating curve, and on the right the related curvature plot.

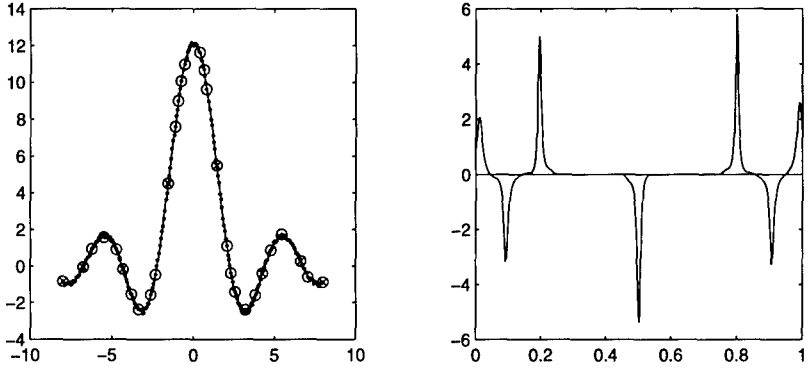


Fig. 2. Test 2: On the left the approximating curve, and on the right the related curvature plot.

In Test 3 the error in the 126 data is obtained by introducing a simulated random perturbation with $d = 0.12$ on both the coordinates of the points $(x_i^e, y_i^e), i = 0, \ldots, 125$ defined as $x_i^e = sin(4\pi \cdot dt \cdot i)$, $y_i^e = cos(2\pi \cdot dt \cdot i), i = 0, \ldots, 125$, where $dt = 0.008$. In this case, among the data there are sequences of quasi-collinear points, and we can observe that the approximating curve has corresponding almost straight line segments.

An application to an engineering problem is presented in Test 4. The 244 data points are derived from measurements effected in the Power Station located in Seraing (Belgium). The measurements are related to the active power of the alternator in the Central, observed on 03/01/1997 between 6:30:00 and 7:00:00. In this case the maximum value of the error is $d = 1$.

It should be noted that, the user needs to work in an interactive way for

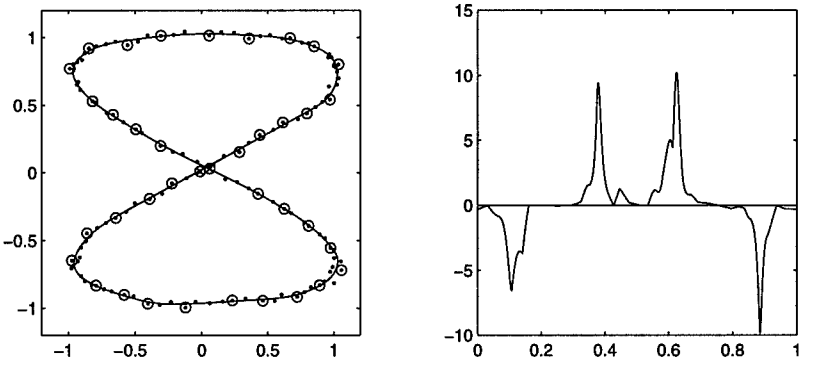


Fig. 3. Test 3: On the left the approximating curve, and on the right the related curvature plot.

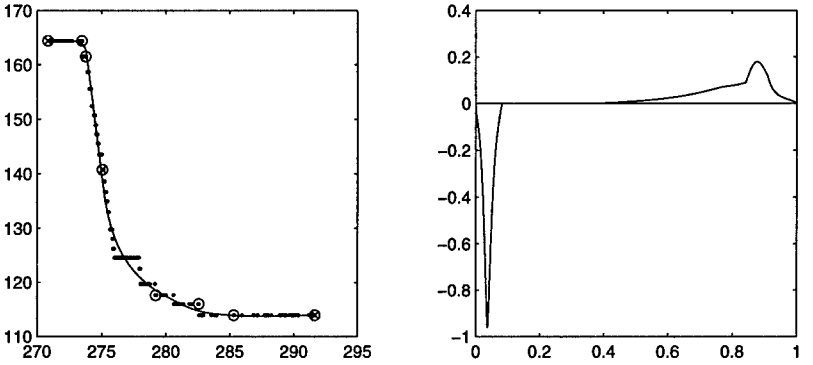


Fig. 4. Test 4: On the left the approximating curve, and on the right the related curvature plot.

selecting suitable values of the input tolerances. Obviously, a previous check of the data shape and of the data distribution is of great help.

§5. Appendix

SD Procedure (Shape Determination)

Input: $\mathcal{P} = \{\boldsymbol{P}_0, \ldots, \boldsymbol{P}_n\}$, d, tol_d, $\mathcal{T} = \{t_0, \ldots, t_n\}$

• Define two auxiliary suitable data points $\boldsymbol{P}_{-1}$ and $\boldsymbol{P}_{n+1}$

if $d = 0$

for $i = 0, \ldots, n$

$$\bullet u_i = \frac{4\Delta_i}{\|\boldsymbol{L}_i\|_2 \cdot \|\boldsymbol{L}_{i+1}\|_2 \cdot \|\boldsymbol{V}_i\|_2}$$

with $\boldsymbol{L}_i = \boldsymbol{P}_i - \boldsymbol{P}_{i-1}$, $\boldsymbol{V}_i = \boldsymbol{P}_{i+1} - \boldsymbol{P}_{i-1}$, $\Delta_i = \frac{1}{2} det(\boldsymbol{L}_i, \boldsymbol{L}_{i+1})$

end

else

$\bullet Lp = \sum_{k=1}^{n-1} ||\boldsymbol{P}_{k+1} - \boldsymbol{P}_k||_2$

for $i = 0, \ldots, n$

$\bullet lev_{r_i} = \max_{j=i+1,\ldots,n+1}\{j : \sum_{k=i}^{j-1} ||\boldsymbol{P}_{k+1} - \boldsymbol{P}_k||_2 < tol_d \cdot Lp\}$

$\bullet lev_{l_i} = \min_{j=-1,\ldots,i-1}\{j : \sum_{k=j}^{i-1} ||\boldsymbol{P}_{k+1} - \boldsymbol{P}_k||_2 < tol_d \cdot Lp\}$

end

for $i = 0, \ldots, n$

• determine the biggest $l_i \in \{-1, \ldots, i-1\}$ and the smallest $r_i \in \{i+1, \ldots, n+1\}$ such that (2) holds replacing $\boldsymbol{P}_{i+1}$ with $\boldsymbol{P}_{r_i}$ and $\boldsymbol{P}_{i-1}$ with $\boldsymbol{P}_{l_i}$

if $l_i < lev_{l_i}$ or $r_i > lev_{r_i}$

$\bullet u_i = 0$

else

$\bullet u_i = \frac{4\tilde{\Delta}_i}{||\tilde{\boldsymbol{L}}_i||_2 \cdot ||\tilde{\boldsymbol{L}}_{i+1}||_2 \cdot ||\tilde{\boldsymbol{V}}_i||_2}$

where $\tilde{\boldsymbol{L}}_i = \boldsymbol{P}_i - \boldsymbol{P}_{l_i}$, $\tilde{\boldsymbol{L}}_{i+1} = \boldsymbol{P}_{r_i} - \boldsymbol{P}_i$, $\tilde{\boldsymbol{V}}_i = \boldsymbol{P}_{r_i} - \boldsymbol{P}_{l_i}$, $\tilde{\Delta}_i = \frac{1}{2} det(\tilde{\boldsymbol{L}}_i, \tilde{\boldsymbol{L}}_{i+1})$

end

end

end

$\bullet i = 1$

$\bullet \Theta_S = \{t_0\}$

while $i < n - 1$

if $u_i = 0$

$\bullet \Theta_S = \Theta_S \cup \{t_{i-1}\}$

• determine $0 \leq id \leq n - i - 1$ such that

$u_i = \ldots = u_{i+id} = 0$ and

$u_{i+id+1} \neq 0$ *or* $id = n - i - 1$

$\bullet \Theta_S = \Theta_S \cup \{t_{\lfloor \frac{2i+id}{2} \rfloor}\}$

$\bullet \Theta_S = \Theta_S \cup \{t_{i+id+1}\}$

$\bullet i = i + id + 2$

elseif $u_i \cdot u_{i+1} < 0$

if $|u_i| < |u_{i+1}|$

$\bullet \Theta_S = \Theta_S \cup \{t_i\}$

$\bullet i = i + 1$

else

$\bullet \Theta_S = \Theta_S \cup \{t_{i+1}\}$

$\bullet i = i + 2$

end

else

$\bullet i = i + 1$

end

end

$\bullet \Theta_S = \Theta_S \cup \{t_n\} \stackrel{\text{def}}{=} \{\tau_0^s, \ldots, \tau_{ns}^s\}$

for $j = 0, \ldots, ns$
  $\bullet s(j) = i$ where $\tau_j^s = t_i$
end
for $j = 0, \ldots, ns-1$
  if $s(j+1) = s(j)+1$
    $\bullet \sigma_j = 0$
  else
    $\bullet \sigma_j = sgn(u_{s(j)+1})$
  end
end
Output: $\mathcal{U} = \{u_0, \ldots, u_n\}$, $\Theta_S = \{\tau_0^s, \ldots, \tau_{ns}^s\}$, $\Sigma = \{\sigma_0, \ldots, \sigma_{ns-1}\}$.

KS Procedure (Knot Selection)

Input:
  $\mathcal{P} = \{\boldsymbol{P}_0, \ldots, \boldsymbol{P}_n\}, \mathcal{T} = \{t_0, \ldots, t_n\}$,
  $\mathcal{U} = \{u_0, \ldots, u_n\}$, $\Theta_S = \{\tau_0^s, \ldots, \tau_{ns}^s\}$,
  L, tol_w, tol_{d1}, tol_{d2} ($tol_{d1} << tol_{d2}$)
$\bullet w_i = |u_i|$, $i = 0, \ldots, n$
$\bullet \Theta = \Theta_S$
$\bullet w_{max} = max\{w_0, \ldots, w_n\}$
for $j = 0, \ldots, ns$
  $\bullet s(j) = i$ where $\tau_j^s = t_i$
end
for $j = 1, \ldots, ns$
  $\bullet s_j = s(j) - s(j-1) + 1$
  • let $\{i_1, \ldots, i_{s_j}\}$ be the index permutation of $\{s(j-1), \ldots, s(j)\}$
  such that the weights $w_{i_1}, \ldots, w_{i_{s_j}}$ are in decreasing order
  $\bullet \Theta_j = \{t_{s(j-1)}, t_{s(j)}\}$
  $\bullet \mathcal{P}_{R_j} = \{\boldsymbol{P}_{s(j-1)}, \boldsymbol{P}_{s(j)}\}$
  for $k = 1, \ldots, s_j$
    if $\frac{w_{i_k}}{w_{max}} > tol_w$ and $\forall \boldsymbol{P}_r \in \mathcal{P}_{R_j}$ $\frac{||\boldsymbol{P}_{i_k} - \boldsymbol{P}_r||_2}{L} > tol_{d1}$
      $\bullet \Theta_j = \Theta_j \cup \{t_{i_k}\}$
      $\bullet \mathcal{P}_{R_j} = \mathcal{P}_{R_j} \cup \{\boldsymbol{P}_{i_k}\}$
    elseif $\frac{w_{i_k}}{w_{max}} \leq tol_w$ and $\forall \boldsymbol{P}_r \in \mathcal{P}_{R_j}$ $\frac{||\boldsymbol{P}_{i_k} - \boldsymbol{P}_r||_2}{L} > tol_{d2}$
      $\bullet \Theta_j = \Theta_j \cup \{t_{i_k}\}$
      $\bullet \mathcal{P}_{R_j} = \mathcal{P}_{R_j} \cup \{\boldsymbol{P}_{i_k}\}$
    end
  end
  $\bullet \Theta = \Theta \cup \Theta_j$
end
Output: $\Theta = \{\tau_0, \ldots, \tau_{nr}\}$

References

1. Arge, E., M. Dæhlen, T. Lyche, and K. Mørken, Constrained spline approximation of functions and data based on constrained knot removal, in *Algorithms for Approximation II*, J. C. Mason and M. G. Cox (eds.), Chapmann and Hall, London, 1990, 4–20.
2. Fletcher, R., *Practical Methods of Optimization*, **2**, John Wiley and Sons, Chichester, 1991.
3. Grace, A., *Optimization Toolbox User's Guide*, The MathWorks, Inc., Natick, Mass, 1994.
4. Hoschek, J., and D. Lasser, *Fundamentals of Computer Aided Geometric Design*, A K Peters, Wellesley, Massachusetts, 1993.
5. Lyche, T., and K. Mørken, Knot removal for parametric B-spline curves and surfaces, Comput. Aided Geom. Design **4** (1987), 217–230.
6. Lyche, T., and K. Mørken, A data reduction strategy for splines with applications to the approximations of functions and data, J. Inst. Math. Applics. **8** (1988), 185–208.

Rossana Morandi
Dip. di Energetica "S. Stecco"
Via Lombroso 6/17
50134 Firenze, Italy
morandi@de.unifi.it

Daniela Scaramelli
Dip. di Matematica Pura ed Applicata
Via G.Belzoni 7
35131 Padova, Italy
daniela@sirio.de.unifi.it

Alessandra Sestini
Dip. di Energetica "S. Stecco"
Via Lombroso 6/17
50134 Firenze, Italy
sestini@de.unifi.it

Convergence of Approximations for Arrangements of Curves

Manuela Neagu and Bernard Lacolle

Abstract. Arrangements of planar objects represent one of the main topics of computational geometry. We propose an approach that allows reliable computations on arrangements of curves. This approach is based on the use of polygonal approximations for the curves composing the arrangement, and the questions to be answered concern the topological and geometric properties of the approximating arrangement of polygonal lines.

§1. Introduction

Problems on arrangements represent one of the most important topics in computational geometry. Arrangements find numerous applications, ranging from the design of 2D drawing tools [9] to motion planning, point location, and visibility problems [5].

Arrangements of hyperplanes, especially arrangements of lines in the plane, have been widely studied. Satisfactory theoretical results (e.g. the zone theorem [7]) and algorithmical results have been found. The interest is now focused on arrangements in higher dimensions, or on planar arrangements of objects other than lines: segments and (Jordan) curves, to give only a few examples. Dealing with arrangements of segments is more difficult than dealing with arrangements of lines because of the larger topological complexity of the cells. The case of curves raises also the fundamental problem of the computation with curves.

Most of the authors that have dealt with curve arrangements have adopted an approach based on the use of a small number of *oracles*. These oracles provide solutions for elementary geometric operations on curves and are considered acurately feasible in constant time. Examples of elementary operations solved by the oracles are the computation of the intersection of two curves (generally, of cardinality greater than 1), the computation of the vertical tangents to a curve, etc. This approach is adopted in papers presenting theoretical results rather than implemented algorithms and numerical results

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 297–306.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

[6,14,15]. Some papers presenting a more application-oriented approach can also be found in the literature [9,12].

In this paper, we present a new approach to the problems of arrangements of curves. We work in a strictly geometric framework, in which the only legal computations are those performed on linear objects. For this purpose, we shall use polygonal approximations of the curves defining the arrangement. Two important questions arise when such an approach is adopted. On the one hand, what kind of information can we obtain on the given arrangement of curves if we avoid algebraic equations? On the other hand, which are the restrictions we must impose on the arrangement of curves in order to assure that the required information can be provided by the arrangement of polygonal approximations?

On an arrangement, there are two different types of results. Firstly, there are topological (or combinatorial) characteristics expressed by the incidence graph. For example, this graph can be used to find the topological closure of a given cell or all its neighbours. Secondly, it can be useful to have geometric information on the faces of the arrangement. That would allow to make a decision for questions such as the point location problem. This paper briefly presents results for both aspects of curve arrangements.

We remark that if we want to compute the incidence graph of an arrangement of curves avoiding algebraic equations, then degenerate cases can not be treated. Indeed, if three curves have a common point, it generally cannot be found via polygonal approximations of the curves. Similarly, if two curves are tangent in a common point, algebraic equations must be generally used to detect the tangency. But on the other hand, our approach provides a robust algorithm for nondegenerate arrangements. Moreover, the method we propose detects the "almost" degenerate positions of the curves. If such a situation occurs, symbolic methods can be employed to obtain the exact local configuration of the arrangement.

The construction of the polygonal approximations is theoretically possible for Jordan curves as general as we want. Practically, the input of an algorithm should be more precise. We have thus chosen to deal with composite Bézier curves. A subject similar to the one of our paper, but concerning only mutually nonintersecting composite Bézier curves, has been treated in [2].

Every curve will be approximated by two polygonal lines: the *control polygon* and the *carrier polygon*. Our approach is thus similar to the one presented in [3,4].

The outline of our paper is the following. In Section 2, we introduce the polycurve, a composite Bézier curve satisfying certain conditions. This is the object we shall deal with throughout the paper. We also define the control and the carrier polygons of a polycurve.

In Section 3, we give basic definitions and notations concerning the simple arrangements of polycurves and control and carrier polygons. In Section 4, we deal with the equivalence of arrangements, for which the definition and sufficient conditions are given. We state the existence of an arrangement of control polygons equivalent to the one of polycurves.

In Section 5, we present our results on the convergence in terms of Hausdorff distance of the cells of the polygonal arrangements to the cells of the curve arrangement. In Section 6, we give a relation of inclusion between the faces of the three arrangements that allows a reliable approach for the point location problem. Finally, Section 7 concludes the paper.

§2. Polycurves

As we can see in the literature [6,12,14,15], the problems on arrangements can be addressed for curves which are subject to very few constraints. But the use of general curves makes the theoretical results unsuitable for direct implementation. This is one of the reasons why we restrict our study to piecewise *completely convex* Bézier curves:

Definition 1. *We say that a Bézier curve is* completely convex *if its control polygon is convex.*

We recall that a curve (or a polygonal chain) is called convex if it is simple and included in the boundary of its convex hull. A completely convex Bézier curve is obviously convex.

Definition 2. *A* polycurve *is a simple curve that can be written as a (finite) union of completely convex Bézier curves.*

An example of polycurve is presented in Figure 1. Let $\mathcal{B} = \cup_{i=1}^{n} \mathcal{B}_i$ be a polycurve, where the control polygon of the Bézier curve $\mathcal{B}_i$ is $\mathcal{P}_i = \mathrm{P}_0^{(i)}\mathrm{P}_1^{(i)} \ldots \mathrm{P}_{m_i}^{(i)}$.

Definition 3. *To any polycurve $\mathcal{B}$ we associate two polygonal chains:*

1) $\mathcal{P} = \cup_{i=1}^{n} \mathcal{P}_i$ *will be called the* bounding polygon *of $\mathcal{B}$;*
2) $\mathcal{S} = \cup_{i=1}^{n} [\mathrm{P}_0^{(i)}\mathrm{P}_{m_i}^{(i)}]$ *will be called the* carrier polygon *of $\mathcal{B}$.*

The bounding and carrier polygons are rough polygonal approximations of the corresponding polycurve. To refine these approximations, we shall apply successive de Casteljau subdivisions to the composing Bézier curves. The subdivision parameter is fixed and equal to $1/2$. On the one hand, this value assures optimal (quadratic) convergence of the control polygon to the associate Bézier curve. On the other hand, the computations are easier and more accurate in this case. Indeed, if the subdivision parameter is equal to $1/2$, then the only arithmetic operations required for the computation of the new control polygons are additions and divisions by 2.

Let us suppose that we subdivide the curve $\mathcal{B}_i$, obtaining the Bézier curves $\mathcal{B}_i'$ and $\mathcal{B}_i''$. Then

$$\begin{aligned}\mathcal{B}_i &= \mathrm{B}_n\big(\mathrm{P}_0^{(i)}, \ldots, \mathrm{P}_{m_i}^{(i)}; [0,1]\big) \\ &= \mathrm{B}_n\big(\mathrm{P'}_0^{(i)}, \ldots, \mathrm{P'}_{m_i}^{(i)}; [0,1]\big) \cup \mathrm{B}_n\big(\mathrm{P''}_0^{(i)}, \ldots, \mathrm{P''}_{m_i}^{(i)}; [0,1]\big) = \mathcal{B}_i' \cup \mathcal{B}_i''.\end{aligned}$$

Thus, $\cup_{j=1}^{n} \mathcal{B}_j$ and $\big(\cup_{j=1}^{i-1} \mathcal{B}_j\big) \cup \mathcal{B}_i' \cup \mathcal{B}_i'' \cup \big(\cup_{j=i+1}^{n} \mathcal{B}_j\big)$ represent the same polycurve. The control and respectively carrier polygons of the two expressions

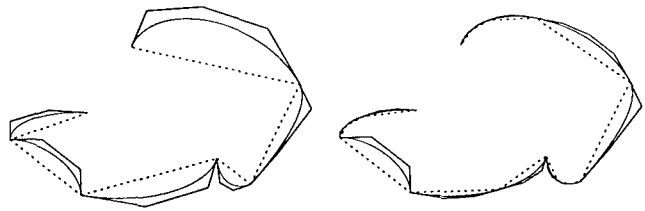


Fig. 1. A polycurve at two different levels of subdivision; the control and carrier polygons are different.

are different: the two polygons associated with a polycurve are not unique, and they change every time one of the composing Bézier curves is subdivided. Figure 1 shows an example of a polycurve with its associate control and carrier polygons before and after the subdivision of some of the composing Bézier curves.

§3. Arrangements of Curves

Let $\Gamma = \{C_i\}_{1 \leq i \leq n}$ be a set of Jordan curves. In this section we give the main definitions related to the arrangement of the curves C_i.

3.1. Arrangement and incidence graph

Definition 4. *The* arrangement $\mathcal{A}(\Gamma)$ *is the planar subdivision induced by the curves of* Γ*; that is,* $\mathcal{A}(\Gamma)$ *is a planar map whose* vertices *are the pairwise intersection points of the curves of* Γ*, whose* edges *are maximal (open) connected portions of the* C_i*'s that do not contain a vertex, and whose* faces *are the connected components of* $\mathbb{R}^2 \setminus \Gamma$.

The vertices, edges, and faces of an arrangement represent its cells of dimension 0, 1, and 2, respectively.

Definition 5. *Let* f *and* g *be two cells of* $\mathcal{A}(\Gamma)$*. If the dimension of* f *is the dimension of* g *plus 1 and* g *is on the boundary of* f*, we say that* g *is a* subcell *of* f *(and* f *is a supercell of* g*). We also say that* f *and* g *define an* incidence*, or are incident to one another.*

Using the previous definition, we can present a useful representation of an arrangement, its incidence graph.

Definition 6. *The* incidence graph *of the arrangement* $\mathcal{A}(\Gamma)$ *is a graph* $G = (V, E)$ *where there is a node in* V *for every cell of* $\mathcal{A}(\Gamma)$*, and an arc between two nodes if the corresponding cells are incident.*

3.2. Simple arrangements

General arrangements of curves can present degeneracies making their study rather tricky. If three curves have a common point, this situation is more difficult to handle than the similar one in the case of arrangements of lines, due to the complexity of the description of the curves. Moreover, two curves can have a common point without crossing at that point (impossible for straight lines). The arrangement is then sensitive to small perturbations.

These are the reasons why, as most of the authors who have studied problems involving arrangements have also done, that we deal exclusively with simple arrangements.

Definition 7. *The arrangement $\mathcal{A}(\mathcal{C})$ is called* simple *if*

1) *the intersection of any three distinct curves C_i, C_j, and C_k is empty;*
2) *if two distinct curves C_i and C_j have common points, they cross transversely in each of these points;*
3) *the set $\cup_{i=1}^{n} C_i$ is connected.*

§4. Topological Approximation

The first question that we answer is: Can we compute the incidence graph of an arrangement of polycurves dealing solely with the polygonal approximations of the polycurves? We have proven that if the arrangement of polycurves is simple, then the answer to this question is yes.

4.1. Equivalence of arrangements

Definition 8 (Grünbaum). *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two arrangements. We say that they are* equivalent *if there exists a bijection $\varphi : \mathcal{A}_1 \longrightarrow \mathcal{A}_2$ such that if f and g define an incidence in $\mathcal{A}_1$, then $\varphi(f)$ and $\varphi(g)$ define an incidence in $\mathcal{A}_2$.*

It is obvious that two arrangements are equivalent if and only if they have the same incidence graph. On the other hand, we remark that if $\mathcal{A}_1$ is simple and $\mathcal{A}_2$ is equivalent to it, this does not imply that $\mathcal{A}_2$ is simple. Property 2 of Definition 7 is not preserved by the equivalence of arrangements.

Let $\mathcal{B} = \{\mathcal{B}_i\}_{1 \leq i \leq n}$ be a set of polycurves. The polycurve $\mathcal{B}_i$ is composed by n_i completely convex Bézier curves $\mathcal{B}_{i,j}$, $\mathcal{B}_i = \cup_{j=1}^{n_i} \mathcal{B}_{i,j}$. $\mathcal{B}_{i,j}$ has degree $m_{i,j}$, and its control polygon will be denoted by $\mathcal{P}_{i,j} = \mathrm{P}_0^{(i,j)} \mathrm{P}_1^{(i,j)} \ldots \mathrm{P}_{m_{i,j}}^{(i,j)}$. We are thus interested in the equivalence of $\mathcal{A}(\mathcal{B})$, $\mathcal{A}(\mathcal{P})$, and $\mathcal{A}(\mathcal{S})$.

Theorem 9. *Let us suppose that $\mathcal{A}(\mathcal{B})$ is simple.*

1) *We can obtain by de Casteljau subdivision a set $\mathcal{P}$ of control polygons and a set $\mathcal{S}$ of carrier polygons of the polycurves of $\mathcal{B}$ such that $\mathcal{A}(\mathcal{P})$ and $\mathcal{A}(\mathcal{S})$ are simple and they remain simple after any further subdivision of the Bézier curves composing the polycurves.*

2) *We can obtain by de Casteljau subdivision a set $\mathcal{P}$ of control polygons and a set $\mathcal{S}$ of carrier polygons of the polycurves of $\mathcal{B}$ such that $\mathcal{A}(\mathcal{P})$ and $\mathcal{A}(\mathcal{S})$ are equivalent to $\mathcal{A}(\mathcal{B})$ and they remain so after any further subdivision of the Bézier curves composing the polycurves.*

We do not present the proof of this theorem here. It is lengthy and presents no technical difficulty. It uses the geometric properties of Bézier curves, namely the variation diminishing property, the inclusion of the curves in the convex hull of its control polygon, and the convergence of the control (and carrier) polygon to the curve by de Casteljau subdivision.

4.2. Polygonal criteria of equivalence

Theorem 9 assures that we can obtain by de Casteljau subdivision arrangements of control, respectively carrier, polygons providing the incidence graph of the corresponding simple arrangement of polycurves. Once these two polygonal arrangements are obtained, the computation of the incidence graph of $\mathcal{A}(\mathcal{B})$ can be done by working solely with polygonal objects. We address now the problem of deciding whether the polygonal arrangements are equivalent to the curve arrangement by performing operations uniquely on linear objects.

Theorem 10. *If the number and the order of all the vertices of $\mathcal{A}(\mathcal{P})$ (respectively $\mathcal{A}(\mathcal{S})$) lying on $\mathcal{P}_i$ (respectively $\mathcal{S}_i$) are the same as the number and the order of all the vertices of $\mathcal{A}(\mathcal{B})$ lying on $\mathcal{B}_i$, for all $i \in \{1, \ldots, n\}$, then $\mathcal{A}(\mathcal{P})$ (respectively $\mathcal{A}(\mathcal{S})$) and $\mathcal{A}(\mathcal{B})$ are equivalent.*

Thanks to the geometric properties of Bézier curves, this theorem is a direct consequence of a result given by Vo Phi [15].

Lemma 11. *We can decide whether the hypotheses of Theorem 10 are fulfilled by dealing solely with the control and the carrier polygons of the polycurves.*

We have established sufficient conditions of equivalence on the two polygonal arrangements. We do not present them here, and just mention that there are two conditions. The first one assures that $\text{card}(\mathcal{B}_{i_1,j_1} \cap \mathcal{B}_{i_2,j_2}) = 1$ when $\text{card}(\mathcal{P}_{i_1,j_1} \cap \mathcal{P}_{i_2,j_2}) = \text{card}(\mathcal{S}_{i_1,j_1} \cap \mathcal{S}_{i_2,j_2}) = 1$, for all $i_1 \neq i_2 \in \{1, \ldots, n\}$ and $j_k \in \{1, \ldots, n_{i_k}\}$, $k = 1, 2$. This implies the equality of the numbers of vertices respectively lying on $\mathcal{P}_i$, $\mathcal{B}_i$ and $\mathcal{S}_i$ in the corresponding arrangements, for all $i \in \{1, \ldots, n\}$. The second condition assures the good ordering of the vertices in the three arrangements.

§5. Approximation in Terms of Distance

Let $\mathcal{B} = \{\mathcal{B}_i\}_{1 \leq i \leq n}$ be a set of polycurves, and let us suppose that $\mathcal{A}(\mathcal{B})$, $\mathcal{A}(\mathcal{P})$, and $\mathcal{A}(\mathcal{S})$ are equivalent. We remark that the problem of the convergence in terms of distance can be addressed also if the three arrangements are not topologically identical, but the discussion is more complex in this case and we do not present it in this paper.

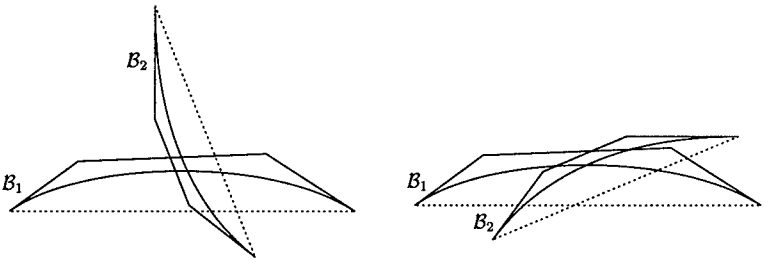


Fig. 2. Different angles between $\mathcal{B}_1$ and $\mathcal{B}_2$ at their intersection point.

5.1. Convergence of vertices

The proof of the following lemma is straightforward.

Lemma 12. *Any vertex of $\mathcal{A}(\mathcal{P})$ (respectively $\mathcal{A}(\mathcal{S})$) converges by subdivision to the corresponding vertex of $\mathcal{A}(\mathcal{B})$.*

We cannot give bounds on the distance between the corresponding vertices of $\mathcal{A}(\mathcal{B})$ and $\mathcal{A}(\mathcal{P})$ (respectively $\mathcal{A}(\mathcal{S})$) depending solely on the Hausdorff distance between the Bézier curves and their control (respectively carrier) polygons. Indeed, it is easy to see that the distance between the intersection point of the curves and the intersection point of the control (respectively carrier) polygons depends on the angle between the curves. An example is presented in Figure 2.

5.2. Convergence of edges

The proof of the following lemma is also straightforward.

Lemma 13. *Let $e_{\mathcal{B}}$ and $e_{\mathcal{P}}$ be corresponding edges of $\mathcal{A}(\mathcal{B})$ and $\mathcal{A}(\mathcal{P})$, and let $v_{\mathcal{B}}, v'_{\mathcal{B}}$ and $v_{\mathcal{P}}, v'_{\mathcal{P}}$ be their corresponding endpoints. There exists $i \in \{1,\dots,n\}$ and $j_1, \ell_1 \in \{1,\dots,m_i\}$, $j_1 \le \ell_1$, such that $e_{\mathcal{B}} \subset \cup_{k=j_1}^{\ell_1} \mathcal{B}_k^{(i)}$. Then*

$$\delta^{\mathrm{H}}(e_{\mathcal{B}}, e_{\mathcal{P}}) \le \max\left\{ \mathrm{d}(v_{\mathcal{B}}, v_{\mathcal{P}}), \mathrm{d}(v'_{\mathcal{B}}, v'_{\mathcal{P}}), \max_{k=j_1}^{\ell_1} \delta^{\mathrm{H}}\left(\mathcal{B}_k^{(i)}, \mathcal{P}_k^{(i)}\right) \right\}.$$

It is obvious that a similar relation holds for the edges of the arrangement of carrier polygons. Lemmas 12 and 13 immediately imply the following statement:

Corollary 14. *Any edge of $\mathcal{A}(\mathcal{P})$ (respectively $\mathcal{A}(\mathcal{S})$) converges by subdivision to the corresponding edge of $\mathcal{A}(\mathcal{B})$.*

5.3. Convergence of faces

The proof of the following lemma is straightforward.

Lemma 15. *Let $c_{\mathcal{P}}$ and $c_{\mathcal{B}}$ be corresponding faces of $\mathcal{A}(\mathcal{P})$ and $\mathcal{A}(\mathcal{B})$. Then*

$$\delta^{\mathrm{H}}(c_{\mathcal{P}}, c_{\mathcal{B}}) \leq \delta^{\mathrm{H}}\left(\delta(c_{\mathcal{P}}), \delta(c_{\mathcal{B}})\right),$$

where $\delta(A)$ denotes the boundary of the set A.

As a matter of fact, this is a more general result, holding for any two compact sets in the plane. Thus, this property is fulfilled also by the faces of the arangement $\mathcal{A}(\mathcal{S})$.

The convergence of faces is an immediate consequence of Lemma 12, Corollary 14, and Lemma 15.

Corollary 16. *Any face of $\mathcal{A}(\mathcal{P})$ (respectively $\mathcal{A}(\mathcal{S})$) converges by subdivision to the corresponding face of $\mathcal{A}(\mathcal{B})$.*

5.4. Polygonal criteria for the convergence in terms of distance

As in Theorem 9 on toplogical convergence of the polygonal arrangements to the curve arrangement, the results we have presented so far in this section imply computations with curves. We give here two results which allow us to estimate the Hausdorff distance between the corresponding cells of $\mathcal{A}(\mathcal{B})$ and $\mathcal{A}(\mathcal{B})$ (respectively $\mathcal{A}(\mathcal{S})$) by performing computations uniquely on polygonal lines. The proofs of the following lemmas are straightforward.

Lemma 17. *If $\mathcal{B}$ is a completely convex Bézier curve and $\mathcal{P} = \mathrm{P}_0\mathrm{P}_1 \ldots \mathrm{P}_m$ is its control polygon, then*

$$\delta^{\mathrm{H}}(\mathcal{P}, \mathcal{B}) \leq \delta^{\mathrm{H}}\left(\mathcal{P}, [\mathrm{P}_0\mathrm{P}_m]\right), \qquad \delta^{\mathrm{H}}\left([\mathrm{P}_0\mathrm{P}_m], \mathcal{B}\right) \leq \delta^{\mathrm{H}}\left(\mathcal{P}, [\mathrm{P}_0\mathrm{P}_m]\right).$$

Lemma 18. *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two completely convex Bézier curves and $\mathcal{P}_i = \mathrm{P}_0^{(i)}\mathrm{P}_1^{(i)} \ldots \mathrm{P}_{m_i}^{(i)}$, $i = 1, 2$, be respectively their control polygons. We suppose that*

$$\operatorname{card}\left(\mathcal{P}_1 \cap \mathcal{P}_2\right) = \operatorname{card}\left(\mathcal{B}_1 \cap \mathcal{B}_2\right) = \operatorname{card}\left([\mathrm{P}_0^{(1)}\mathrm{P}_{m_1}^{(1)}] \cap [\mathrm{P}_0^{(2)}\mathrm{P}_{m_2}^{(2)}]\right) = 1.$$

Then

$$\mathrm{d}\left(\mathcal{P}_1 \cap \mathcal{P}_2, \mathcal{B}_1 \cap \mathcal{B}_2\right) \leq \mathrm{d}\left(\mathcal{P}_1 \cap \mathcal{P}_2, [\mathrm{P}_0^{(1)}\mathrm{P}_{m_1}^{(1)}] \cap [\mathrm{P}_0^{(2)}\mathrm{P}_{m_2}^{(2)}]\right)$$

and

$$\mathrm{d}\left([\mathrm{P}_0^{(1)}\mathrm{P}_{m_1}^{(1)}] \cap [\mathrm{P}_0^{(2)}\mathrm{P}_{m_2}^{(2)}], \mathcal{B}_1 \cap \mathcal{B}_2\right) \leq \mathrm{d}\left(\mathcal{P}_1 \cap \mathcal{P}_2, [\mathrm{P}_0^{(1)}\mathrm{P}_{m_1}^{(1)}] \cap [\mathrm{P}_0^{(2)}\mathrm{P}_{m_2}^{(2)}]\right).$$

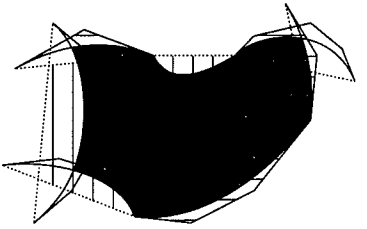


Fig. 3. $c_{\mathcal{B}}$ is not included in $c_{\mathcal{P}} \cap c_{\mathcal{S}}$.

§6. Inclusion of Faces

The results given in Section 5 can be very useful for solving the point location problem in the case of curve arrangements. In this section, we present a result that shows how the convergence in terms of distance of the faces of $\mathcal{A}(\mathcal{P})$ and respectively $\mathcal{A}(\mathcal{S})$ to the faces of $\mathcal{A}(\mathcal{B})$ is applied to the mentioned problem. We use for the set of polycurves $\mathcal{B}$ the notations of Section 4, that is, $\mathcal{B} = \{\mathcal{B}_i\}_{1 \leq i \leq n}$, where $\mathcal{B}_i = \cup_{j=1}^{n_i} \mathcal{B}_{i,j}$ with $\mathcal{B}_{i,j}$ completely convex Bézier curves for all i and j.

Lemma 19. *Let $c_{\mathcal{B}}$ be a face of $\mathcal{A}(\mathcal{B})$, and $c_{\mathcal{P}}$, $c_{\mathcal{S}}$ be the faces that correspond to $c_{\mathcal{B}}$ in $\mathcal{A}(\mathcal{P})$, respectively $\mathcal{A}(\mathcal{S})$. Let $v_1, \ldots, v_p$ be all the vertices of $\mathcal{A}(\mathcal{B})$ lying on the boundary of $c_{\mathcal{B}}$. There exist $i_k \neq i'_k \in \{1, \ldots, n\}$ and $j_k \in \{1, \ldots, n_i\}$, $j'_k \in \{1, \ldots, n_{i'}\}$, for all $k \in \{1, \ldots, p\}$, such that $v_k = \mathcal{B}_{i_k,j_k} \cap \mathcal{B}_{i'_k,j'_k}$. Then*

$$c_{\mathcal{P}} \cap c_{\mathcal{S}} \subset c_{\mathcal{B}} \subset c_{\mathcal{P}} \cup c_{\mathcal{S}} \cup \bigcup_{k=1}^{p} \left(reg(\mathcal{P}_{i_k,j_k}) \cap reg(\mathcal{P}_{i'_k,j'_k}) \right),$$

where $reg(\mathcal{P}_{i,j})$ denotes the bounded region enclosed by the polygon $\mathcal{P}_{i,j}$.

This property is illustrated in Figure 3. We remark that in fact the terms $reg(\mathcal{P}_{i_k,j_k}) \cap reg(\mathcal{P}_{i'_k,j'_k})$ are not all necessary. When the Bézier curves $\mathcal{B}_{i_k,j_k}$ and $\mathcal{B}_{i'_k,j'_k}$ both have their "convex side" oriented either to the interior of the face $c_{\mathcal{B}}$ or to the exterior of this face, we do not have to add $reg(\mathcal{P}_{i_k,j_k}) \cap reg(\mathcal{P}_{i'_k,j'_k})$ to the union above.

§7. Conclusion

In this paper, we briefly present results concerning the use of polygonal approximations for solving important computational geometry problems on arrangements of curves. We have dealt with two different types of problems, topological and geometric. For both kinds of problems, the polygonal approximations represent a suitable tool, providing solutions that do not require solving algebraic equations.

References

1. Calcoen, E., Approximations polygonales d'objets convexes du plan pour la géométrie algorithmique, Ph.D. Thesis, University Joseph Fourier, Grenoble, 1996.
2. Cho, W., T. Maekawa, and N. M. Patrikalakis, Topological reliable approximation of composite Bézier curves, Comput. Aided Geom. Design **13** (1996), 497–520.
3. Dobkin, D. P., and D. L. Souvaine, Computational geometry in a curved world, Algorithmica **5** (1990), 421–457.
4. Dobkin, D. P., D. L. Souvaine, and C. J. Van Wyk, Decomposition and intersection of simple splinegons, Algorithmica **3** (1988), 473–485.
5. Edelsbrunner, H., *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.
6. Edelsbrunner, H., L. Guibas, J. Pach, R. Seidel, and M. Sharir, Arrangements of curves in the plane – topology, combinatorics, and algorithms, Theoret. Comput. Sci. **92** (1992), 319–336.
7. Edelsbrunner, H., J. O'Rourke, and R. Seidel, Constructing Arrangements of Lines and Hyperplanes with Applications, SIAM J. Comput. **15** (1986), 341-363.
8. Farin, G., *Curves and Surfaces for Computer Aided Geometric Design: a Practical Guide*, London Academic Press, 1997.
9. Gangnet, M., J.-C. Hervé, T. Pudet, and J.-M. Van Thong, Incremental computation of planar maps, Computer graphics **23**, no. 3 (1989), 345–354.
10. Grünbaum, B., *Arrangements and Spreads*, AMS, Providence, RI, 1972.
11. Halperin, D., Arrangements, in *Handbook of Discrete and Computational Geometry*, CRC Press LLC, Boca Raton, 1997, 389–412.
12. Milenkovic, V., Calculating approximate curve arrangements using rounded arithmetic, in Proc. 5th Annu. ACM Sympos. on Comput. Geom. (1989), 197–207.
13. Sharir, M., and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, 1995.
14. Snoeyink, J., and J. Hershberger, Sweeping arrangements of curves, in Proc. 5th Annu. ACM Sympos. on Comput. Geom. (1989), 354–363.
15. Vo Phi, K., Contributions à l'étude des arrangements : équivalences combinatoires et perturbations, Ph.D. Thesis, University Joseph Fourier, Grenoble, 1994.

Manuela Neagu and Bernard Lacolle
LMC-IMAG, BP 53
38041 Grenoble Cedex 9, France
{Manuela.Neagu, Bernard.Lacolle}@imag.fr

The (2-5-2) Spline Function

Jae H. Park and Leonard A. Ferrari

Abstract. Splines have been used extensively in the interpolation of multidimensional data sets. Linear interpolation utilizes second order splines (first degree piecewise polynomials) and has widespread popularity because of its ease of implementation. Cubic splines are often used when higher degrees of smoothness are required of the interpolation process. Linear interpolation has the advantages of not requiring the solution of an inverse problem (the data points are themselves the coefficients of the triangular basis functions) and extremely efficient generation of the output sample points. Unfortunately, the linear-interpolating function has only C^0 continuity (the function is continuous but its derivatives are discontinuous) and therefore lacks the required smoothness for many applications. We provide a new algorithm in this paper based on the efficient derivative summation approach to spline rendering. Cubic B-spline interpolation for uniformly spaced data points provides C^2 continuity. The interpolation function can be rendered quite efficiently from the basis coefficients and the basis function, using a cascade of four running average filters. Unser *et al.* have shown a digital filter solution for the inverse problem of obtaining the spline coefficients from the data points. A matrix inversion solution is also commonly used. Both solutions require the use of floating point multiplication and addition, while the forward problem can be implemented utilizing only fixed-point additions. In this paper, we develop a class of spline basis functions which solve the interpolation problem using only simple arithmetic shifts and fixed point additions for solutions to both the forward and inverse problems. The system impulse response for the new interpolators appears to be closer to the ideal interpolator than the B-spline interpolator. We refer to the new splines as (2-5-2) splines

§1. Introduction

Splines are well-accepted in Computer Graphics[1,2,3,4]. The high computational requirements of cubic splines and the large amounts of data make them difficult to use in multi-dimensional applications. In many applications, bilinear interpolation is used instead of cubic splines because of its simplicity in implementation. However, bilinear interpolation cannot produce images of sufficient quality for many application because of its C^0 continuity. The

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 307–314.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

(2-5-2) spline approach can be computed at nearly the same efficiency as the bilinear interpolator, but provides much higher quality results.

In general, spline computation can be divided into two parts. The first part requires the solution of an inverse problem to obtain the coefficients (vertices) of the basis functions, and the second part is a forward computation to generate the interpolating spline from its coefficients.

§2. Inverse Problem

Assume we are given $(m-r+3)$ data values P_i, $i = r-1, \ldots, m+1$. Further assume that we wish to find a continuous spline curve such that on a specified set of equally spaced points, $\bar{u}_i$ $i = r-1, \ldots, m+1$, the curve attains the values P_i. Let $B_{i,r}(\bar{u})$ denote an rth order set of basis splines defined on the knot set $i = 0, \ldots, m$. Then, for the case of a curve in one-dimension we can write

$$Q(\bar{u}_p) = \sum_{i=0}^{m} V_i B_{i,r}(\bar{u}_p) = P_p, \qquad p = k-1, \ldots, m+1. \tag{1}$$

This represents $m-r+3$ equations in $m+1$ unknowns. In the case of a cubic spline, $r = 4$ and we are short two equations. We are free to augment the data set P_i at both ends with additional data values. However, these augmented data values effect the shape of the interpolating function $Q(\bar{u})$ non-locally.

In matrix form, we obtain

$$\begin{bmatrix} B_{0,r}(\bar{u}_{r-2}) & \cdots & B_{m,r}(\bar{u}_{r-2}) \\ B_{0,r}(\bar{u}_{r-1}) & & B_{m,r}(\bar{u}_{r-1}) \\ \vdots & & \vdots \\ B_{0,r}(\bar{u}_{m+2}) & \cdots & B_{m,r}(\bar{u}_{m+2}) \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ \vdots \\ V_m \end{bmatrix} = \begin{bmatrix} P_{r-2} \\ P_{r-1} \\ \vdots \\ P_{m+2} \end{bmatrix} \tag{2}$$

or

$$B\underline{V} = \underline{P},$$

where $\underline{V}$ denotes the vector of unknown coefficients, B represents the matrix of basis spline values at the knot locations and $\underline{P}$ represents the given set of data points including the augmented values. The solution to (2) exists whenever the matrix B has an inverse. The solution to (2) is efficient whenever $B^{-1}\underline{P}$ or its equivalent is easy to compute.

For the cubic B-spline defined on uniform knots, it is easy to show that the interpolation problem utilizing simple, uniform knot B-splines leads to the inversion problem

$$\frac{1}{6}\begin{bmatrix} 4 & 1 & 0 & . & . & 0 \\ 1 & 4 & 1 & 0 & . & 0 \\ 0 & 1 & 4 & 1 & . & 0 \\ . & . & . & . & . & . \\ 0 & . & 0 & 1 & 4 & 1 \\ 0 & . & . & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} V_0 \\ . \\ . \\ . \\ . \\ V_m \end{bmatrix} = \begin{bmatrix} P_2 \\ . \\ . \\ . \\ . \\ P_{m+2} \end{bmatrix}. \tag{3}$$

In this case the matrix B can be inverted using forward elimination and backwards substitution as in [1]. While the B-spline matrix is banded and can therefore be inverted reasonably efficiently, the process requires full floating point multiplication and addition. Even after the matrix inversion, to obtain the vertices, full floating point multiplication and addition must be used.

The first attempt at reducing the number of the calculations was made by Unser, *et al* with a digital filtering solution to the problem by noting that the inverse of the FIR filter given by $B(z) = z + 4 + z^{-1}$ is the filter with impulse response [3]

$$b^{-1}(n) = \frac{-6\alpha}{(1-\alpha^2)}\alpha^{|n|}, \tag{4}$$

where, $\alpha = \sqrt{3} - 2$ is the smallest root in absolute value of the polynomial $z^2 + 4z + 1$.

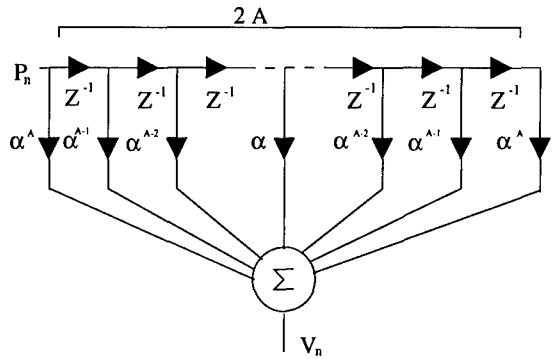
As they implemented it, the above IIR filter should be split into causal and non-causal parts, and applied twice to obtain the vertices: the non-causal sequences and anti-causal sequences. Since $b^{-1}(n)$ becomes smaller as $|n|$ gets larger, we can assume the filter has only several non-zero elements from the center of the filter sequences. Thus, we can approximate the filter with an FIR filter as below, and call it the Inversion FIR filter:

$$b^{-1}(n) = \begin{cases} \frac{-6\alpha}{(1-\alpha^2)}\alpha^{|n|}, & \text{if } |n| \leq m, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Even with the Inversion FIR filter, we still need full floating point multiplication and addition to get the vertices. The (1-4-1) spline does not provide a simple solution to the coefficient inverse problem because its characteristic polynomial, $z^2 + 4z + 1$, has irrational roots. The polynomials $2z^2 + 5z + 2$ has roots which are negative powers of two. We refer to these splines as (2-5-2) splines which have roots which are of similar magnitude to the roots of the (1-4-1) spline.

We assume the spline is defined on the uniform set $\{-2, -1, 0, 1, 2\}$, and that it takes on the set of values $\{0, \frac{2}{9}, \frac{5}{9}, \frac{2}{9}, 0\}$ at the knots. In Sect. 6, we set up sixteen equations to solve for the 16 polynomial coefficients defining the (2-5-2) spline. We note that it has C^2 continuity at knots 1, 2 and 3 and C^1 continuity at the knots at 0 and 4 (from the polynomials). Hence, the (2-5-2) spline is a multiple knot spline defined on seven knots with the knots at 0 and 4 having multiplicity two. We also note that the normalization property holds, that is $\sum_{i=0}^{3} P_i(\bar{u}) = 1$, for $u \in [0, 1]$.

If, as in [3], we assume periodic boundary conditions, equation (2) takes the following circulant form:



A : # of bits in the fixed point number

V_n: Vertices

P_n: Sampled data

Fig. 1. The inverse filter for a (2-5-2) spline: $\alpha = -1/2$.

$$\frac{1}{9}\begin{bmatrix} 5 & 2 & 0 & . & . & 0 & 2 \\ 2 & 5 & 2 & 0 & . & . & 0 \\ 0 & 2 & 5 & 2 & . & . & 0 \\ 0 & . & . & . & . & . & . \\ . & . & . & . & 2 & 5 & 2 \\ 2 & 0 & . & . & 0 & 2 & 5 \end{bmatrix} \begin{bmatrix} V_0 \\ . \\ . \\ . \\ . \\ V_m \end{bmatrix} = \begin{bmatrix} P_2 \\ . \\ . \\ . \\ . \\ P_{m+2} \end{bmatrix} \tag{6}$$

or

$$\frac{1}{9} B_5 \underline{V} = \underline{P}. \tag{7}$$

With the approach Unser, *et al* applied to (1-4-1) splines, and the FIR filter approximation, we can obtain a Inversion FIR filter for (2-5-2) splines and the FIR filter will be

$$b^{-1}(n) = \begin{cases} 3\left(-\frac{1}{2}\right)^{|n|}, & \text{if } |n| \le m, \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The filter can be implemented as shown in Fig. 1 with delays, mulipliers and adders. Although the filter shown in Fig. 1 can be used for a (1-4-1) spline with change of $\alpha = \sqrt{3} - 2$, the FIR filter for a (2-5-2) spline can be implemented with an integer processer because all its coefficients are powers of 1/2, and the power of two multiplication can be realized with shifts.

§3. Forward Problem

Ferrari, *et al.*, provided an efficient algorithm (the Fast Spline Transform, (FST)) for computing a spline by Derivative/Summation [5]. Once we obtain

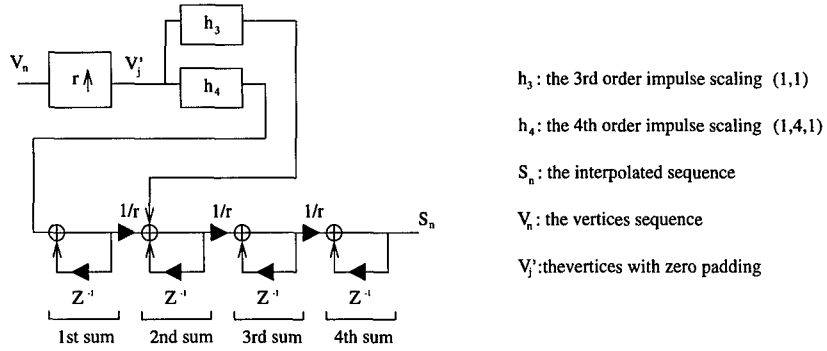


Fig. 2. The FST filter for (2-5-2) spline calculation.

the vertices, the spline is calculated by the FST with appropriate Impulse Scaling. The FST is as follows, where r is the order of the basis function and m is the interpolation ratio:

Algorithm FST

1) Find the Dirac functions corresponding to all orders of the rth order spline's derivatives $(r, r-1, \cdots, 1)$ [4].
2) Create an array of zeros with $m-1$ elements between the knot locations; initialize $k = r$.
3) Scale the appropriate Dirac functions by amplitude V_n, and sum the resulting sequence into the array at the knot locations corresponding to the Dirac functions(requires shift and add for the (2-5-2) spline).
4) "Integrate" the resulting sequence once using repeated summation. Set $k = k - 1$.
5) Return to step 3. Until $k = 0$.
6) The array contains the values of the spline at the specified locations.

The FST algorithm can be implemented as a digital filter for any spline. However, to implement this with only fixed point shifts and additions, every Impulse Scaling element(coefficient) of the spline must be powers of two. The suggested filter (the FST filter) for the (2-5-2) spline is shown in Fig. 2. Because the (2-5-2) spline has double knots, both 4th and 3rd order impulse sets exist. h_4 and h_3 in Fig. 2 correspond to 4th order impulse scaling and 3rd order impulse scaling. Fig. 2 shows clearly that the forward computation of the spline is computed at the input data rate.

§4. Cosine Examples

Since the Impulse Scaling approach for the forward problem always generates any (2-5-2) spline curves on the defined grids, the Impulse Scaling will not effect the accuracy of the generated curves by each Inverse Problem scheme

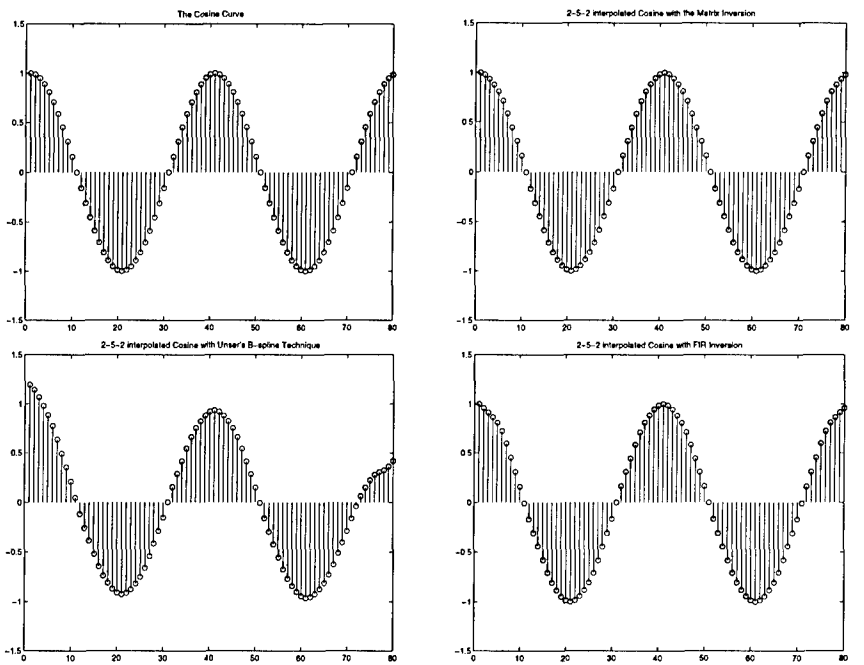


Fig. 3. The cosine interpolated examples: Top Left: True Cosine Curve, Top Right: by Matrix Inversion, Bottom Left: by Unser's Inversion and by FIR inversion.

(i.e., Matrix Inversion, Unser's IIR Filter Inversion and FIR Filter Inversion). The inversion schemes affect it. Unser's IIR filter Inversion and FIR filter Inversion can be considered as an approximation operation of matrix inversion. As shown in Fig. 3, the cosine curve generated by matrix inversion is closest to the actual cosine curves because (2) guarantees that the (2-5-2) spline curve passes through every data point (P_n's). Although the two filter approaches are not guaranteed to pass through every data point, they all produce fairly accurate curves in the middle section. For the (2-5-2) spline, they can be implemented with an integer process because the Filter coefficients of IIR filter and FIR filter's are powers of two. It appears that FIR filter inversion generates more accurate cosine than Unser's IIR filter Invesion in Fig. 3, while the FIR filter is nothing but the approximation of the IIR filter. The initial value estimation for the anti-causal realization of the IIR filter results in less accuracy. It can be confirmed that significant distortion appears at the right end portion of the IIR filter inversion cosine interpolated curve in Fig. 3.

§5. Discussion

With either FIR Filter Inversion or IIR Filter Inversion, the (2-5-2) spline interpolation will generates more accurate curves or surfaces than the (1-4-1)

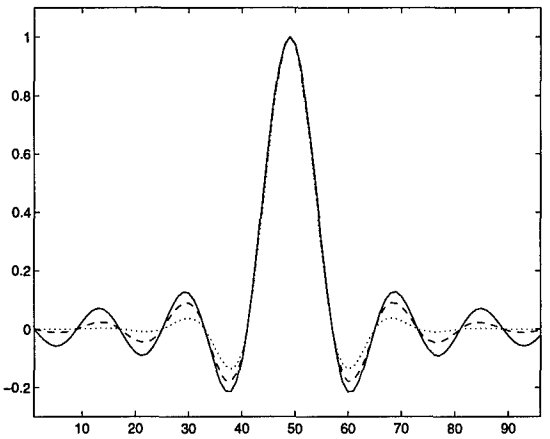


Fig. 4. The impulse responses of a (1-4-1) spline(the dotted line), a (2-5-2) spline(the dashed line) and an ideal interpolator(the solid line).

spline interpolation does. This can be easily confirmed by the fact that the (2-5-2) spline's impulse response is much closer to the sinc function than that of the (1-4-1) spline (refer to Fig. 4). Because the sinc function is the base of the perfect interpolation by the sampling therom, the (2-5-2) spline produces more accurate interpolated results than the B-spline. In addition, the (2-5-2) can be implemented with only fixed point shifts and additions. Therefore, the (2-5-2) spline is at least as accurate as the B-spline, and can be computed much more efficiently.

§6. Polynomial Coefficients for the (2-5-2) Spline

The spline is defined on five uniformly spaced knots $\{\bar{u_0}, \bar{u_1}, \bar{u_2}, \bar{u_3}, \bar{u_4}\}$. For each interval, we define $u \in [0,1]$ as the polynomial variable. Then for each interval, the coefficients $\{a_i, b_i, c_i, d_i\}$ represent the polynomial $a_i + b_i u + c_i u^2 + d_i u^3$. We denote the four polynomials by $p_0(u), p_1(u), p_2(u)$, and $p_3(u)$.

We impose the following sixteen constraints:

i. $P_0(0) = 0$

ii. $P_3(1) = a_3 + b_3 + c_3 + d_4 = 0$

iii. $P_0'(0) = b_0 = 0$

iv. $P_3'(0) = b_3 + 2c_3 + 3d_3 = 0$

v. $P_0(1) = a_0 + b_0 + c_0 + d_0 = \frac{2}{9}$

vi. $P_1(0) = a_1 = \frac{2}{9}$

vii. $P_2(1) = a_2 + b_2 + c_2 + d_2 = \frac{2}{9}$

viii. $P_3(0) = a_3 = \frac{2}{9}$

ix. $P_1(1) = a_1 + b_1 + c_1 + d_1 = \frac{5}{9}$

x. $P_2(0) = a_2 = \frac{5}{9}$

xi. $P_0'(1) = b_0 + 2c_0 + 3d_0 = P_1'(0) = b_1$

xii. $P_1'(1) = b_1 + 2c_1 + 3d_1 = P_2'(0) = b_2$

xiii. $P_2'(1) = b_2 + 2c_2 + 3d_2 = P_3'(0) = b_3$

xiv. $P_0''(1) = 2c_0 + 6d_0 = P_1''(0) = 2c_1$

xv. $P_1''(1) = 2c_1 + 6d_1 = P_2''(0) = 2c_2$

xvi. $P_2''(1) = 2c_2 + 6d_2 = P_3''(0) = 2c_3$

References

1. Bartels, R., J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling.*, Morgan Kaufman publishers, 1987, Los Altos, California.
2. de Boor, C., *A Practical Guide to Splines*, Applied Mathematical Sciences, Vol. 27, Springer-Verlag, New York, 1978.
3. M. Unser, A. Aldroubi, and M. Eden, Fast B-Spline transforms for continuous image representation and interpolation, IEEE Transactions on Pattern Analysis and Machine Intelligence, **13**, No.3, March (1991), 277–285.
4. Sankar, P., M. Silbermann, and L. Ferrari, Curve and surface generation and refinement based on a high speed derivative algorithm, CVGIP: Graphical Models and Image Processing, **56**, No. 1, (1994), 94–101.
5. Ferrari, L. A., J. H. Park, A. Healey and S. Leeman, Interpolation using fast spline transform (FST), IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, **46**, (1999), 891–906

Jae H. Park and Leonard A. Ferrari
The Bradley Department of Electrical Computer Engineering
340 Whittemore Hall
Virginia Tech
Blacksburg, VA 24061-0111
ferrari@vt.edu

Error Analysis of Algorithms for Evaluating Bernstein-Bézier-Type Multivariate Polynomials

J. M. Peña

Abstract. In Computer Aided Geometric Design, the Bernstein-Bézier form is the usual way to store a polynomial defined on a triangle. We perform backward and forward error analysis of the de Casteljau algorithm and of the algorithm proposed by Schumaker and Volk for evaluating such polynomials. The obtained results are also compared with the corresponding results for the bivariate Horner algorithm.

§1. Introduction

In Computer Aided Geometric Design, multivariate polynomials defined on a triangle are usually stored in the Bernstein-Bézier form, and can be evaluated by the de Casteljau algorithm. In [8] a modified Bernstein-Bézier representation of polynomials was introduced, along with an algorithm for its evaluation, which was called the VS algorithm. This algorithm for the evaluation of multivariate polynomials is expressed in terms of nested multiplications, and is more efficient than the de Casteljau algorithm.

Error analysis of the de Casteljau algorithm for univariate polynomials was considered in [2] and [4]. This paper is devoted to backward and forward error analysis of the de Casteljau and VS algorithms for bivariate polynomials. On the other hand, the error analysis of the Horner algorithm for the evaluation of univariate polynomials has been extensively studied in the literature. In fact, backward and forward error analysis of (univariate) Horner's rule was already performed by Wilkinson in [11], pp. 36–37 and 49–50. Other approaches to this problem can be found in [5,9,10,12] (see more references in [3]). An error analysis of the multivariate Horner algorithm has been given in [7]. We also compare our results with those corresponding to the multivariate Horner algorithm.

The paper is organized as follows. Section 2 introduces basic concepts, notation, and auxiliary results. In Section 3 we carry out the mentioned error analysis of the algorithms. Finally, we summarize in Section 4 the main conclusions and the advantages of VS algorithm in this context, taking into account computational cost and forward error analysis.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 315–324.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

§2. Basic Notations and Auxiliary Results

Let us now introduce some standard notation in error analysis. Given $a \in \mathbb{R}$, the computed element in floating point arithmetic will be denoted by either $\mathrm{fl}(a)$ or by $\hat{a}$. As usual, to investigate the effect of rounding errors when working with floating point arithmetic, we use the model

$$\mathrm{fl}(a \,\mathrm{op}\, b) = (a \,\mathrm{op}\, b)(1+\delta), \quad |\delta| \leq u, \tag{1}$$

with u the unit roundoff and *op* any of the elementary operations $+$, $-$, $\times$, $/$ (see [3, p. 44] for more details). Given $k \in \mathbf{N}_0$ such that $ku < 1$, let us define

$$\gamma_k := \frac{ku}{1-ku}. \tag{2}$$

In our error analysis we shall deal with quantities such that their absolute value is bounded above by γ_k. Following [3], we denote such quantities by θ_k and take into account that by Lemmas 3.3 and 3.1 of [3], the following properties hold:

$$(1+\theta_k)(1+\theta_j) = 1+\theta_{k+j}, \tag{3}$$

and if $\rho_i = \pm 1$, $|\delta_i| \leq u$ $(i = 1, \ldots, k)$ then

$$\prod_{i=1}^{k}(1+\delta_i)^{\rho_i} = 1+\theta_k. \tag{4}$$

In considering the computed solution of a problem, one can try to find the data for which this computed solution is the exact solution. Backward error analysis measures how far these data are from the original data of the problem. So, backward error analysis interprets rounding errors as perturbations in the data. In contrast, forward error analysis measures how far the computed solution is from the exact solution. Therefore, in our evaluation problem, if $\hat{f}(x) = \sum_{i=0}^{n} \bar{c}_i u_i(x)$ is the computed evaluation (instead of the exact evaluation $f(x) = \sum_{i=0}^{n} c_i u_i(x)$), we say that the relative backward error is bounded above by ε if

$$\frac{|\bar{c}_i - c_i|}{|c_i|} \leq \varepsilon, \quad i = 0, \ldots, n.$$

Then we can bound the forward error by

$$|\hat{f}(x) - f(x)| \leq \varepsilon \sum_{i=0}^{n} |c_i u_i(x)|.$$

The number

$$C_u(f(x)) := \sum_{i=0}^{n} |c_i u_i(x)|, \tag{5}$$

measures the stability in the evaluation of a function with respect to perturbations of the coefficients, and is called the condition number for the evaluation of

$f(x)$ with the basis u (see [1,2,4,6,7]). Let us observe that $C_u(f(x))$ depends on the basis u, on the function f, and on the point x. If we assume that the basis is formed by nonnegative functions, (5) can be written as

$$C_u(f(x)) = \sum_{i=0}^{n} |c_i| u_i(x). \tag{6}$$

In conclusion, we can bound the forward error by

$$|\hat{f}(x) - f(x)| \leq \varepsilon C_u(f(x)), \tag{7}$$

which is a particular case of the classical formula *(Forward error)* $\leq$ *(Backward error)* $\times$ *(Condition number).*

If we assume that $f(x) \neq 0$ we can also define the relative condition number by

$$c_u(f(x)) := \frac{C_u(f(x))}{|f(x)|}.$$

The following relative forward error bound, analogous to (7), can be derived:

$$\frac{|\hat{f}(x) - f(x)|}{|f(x)|} \leq \varepsilon c_u(f(x)).$$

The following result (which was obtained for polynomials in [1]) allows us to compare the condition number of two bases of nonnegative functions in a space when they are related by a nonnegative matrix:

Lemma 1. *Let $\mathcal{U}$ be a finite dimensional vector space of functions defined on $\Omega \subset \mathbb{R}^s$. Let $u = (u_0, \ldots, u_n)$, $v = (v_0, \ldots, v_n)$ be two bases of nonnegative functions of $\mathcal{U}$ such that*

$$(v_0, \ldots, v_n) = (u_0, \ldots, u_n)A,$$

where $A = (a_{ij})_{0 \leq i,j \leq n}$ is a nonnegative matrix. Then $C_u(f(x)) \leq C_v(f(x))$ for each function $f \in \mathcal{U}$ evaluated at every $x \in \Omega$.

Proof: Given $f \in \mathcal{U}$, it can be written as

$$f(x) = \sum_{q=0}^{n} c_q v_q(x) = \sum_{i=0}^{n} \left(\sum_{q=0}^{n} (c_q a_{iq}) \right) u_i(x). \tag{8}$$

Then, by (8) and the nonnegativity of u, v and A, we deduce that

$$\begin{aligned} C_u(f(x)) &= \sum_{i=0}^{n} \Big| \sum_{q=0}^{n} (c_q a_{iq}) \Big| |u_i(t)| \\ &\leq \sum_{i=0}^{n} \sum_{q=0}^{n} |c_q| a_{iq} u_i(x) = \sum_{q=0}^{n} |c_q| \left(\sum_{i=0}^{n} a_{iq} u_i(x) \right) \\ &= \sum_{q=0}^{n} |c_q| \, |v_q(x)| = C_v(f(x)) \end{aligned} \tag{9}$$

for each function $f \in \mathcal{U}$ evaluated at every $x \in \Omega$. $\square$

§3. Backward and Forward Error Analysis of the Algorithms

Given a triangle T with vertices P, Q, R in the plane and a point $X \in T$, let (r, s, t) be the barycentric coordinates of X:

$$X = rP + sQ + tR, \quad r + s + t = 1.$$

Let $\Pi_d(T)$ space of polynomials of total degree d defined on T. Then any polynomial $p \in \Pi_d(T)$ can be written as

$$p(r, s, t) = \sum_{i=0}^{d} \sum_{j=0}^{i} b_{d-i,i-j,j} B^d_{d-i,i-j,j}(r, s, t), \tag{10}$$

where

$$B^d_{i,j,k}(r, s, t) = \frac{d!}{i!j!k!} r^i s^j t^k, \quad i + j + k = d \tag{11}$$

are the Bernstein polynomials. Then (10) is called the Bernstein-Bézier representation of p.

Let us now recall the algorithm of de Casteljau to evaluate the polynomial p at a given (r, s, t). We denote by $b^0_{i,j,k} := b_{i,j,k}$ for all $i + j + k = d$.

Algorithm of de Casteljau

for $k = 1$ **to** d
  for $i = 0$ **to** $d - k$
    for $j = 0$ **to** i
      $b^k_{d-i-k,i-j,j} = r \cdot b^k_{d-i-k+1,i-j,j} + s \cdot b^{k-1}_{d-i-k,i-j+1,j} + t \cdot b^{k-1}_{d-i-k,i-j,j+1}$
    end j
  end i
end k

The previous algorithm leads to the evaluation $p(r, s, t) = b^d_{0,0,0}$. It requires $d(d + 1)(d + 2)/2$ multiplications.

The following result provides backward error analysis of the de Casteljau algorithm for the evaluation of bivariate polynomials.

Theorem 2. *Let us consider the algorithm of de Casteljau in (10) and let us assume that* $3du < 1$. *Then the computed value* $\hat{p}(r, s, t) = \mathrm{fl}(p(r, s, t))$ *satisfies*

$$\hat{p}(r, s, t) = \sum_{i=0}^{d} \sum_{j=0}^{i} \bar{b}_{d-i,i-j,j} B^d_{d-i,i-j,j}(r, s, t), \tag{12}$$

where

$$\frac{|\bar{b}_{i,j,k} - b_{i,j,k}|}{|b_{i,j,k}|} \leq \gamma_{3d}. \tag{13}$$

Proof: By (1), for each $k \in \{1, \ldots, d\}$

$$\begin{aligned}
&\hat{b}^k_{d-i-k,i-j,j} \\
&\quad = \left[\mathrm{fl}(r b^k_{d-i-k+1,i-j,j}) + \mathrm{fl}(s b^{k-1}_{d-i-k,i-j+1,j} + t b^{k-1}_{d-i-k,i-j,j+1})\right](1+\delta_0) \\
&\quad = \left[(r\hat{b}^k_{d-i-k+1,i-j,j})(1+\delta_1) + (\mathrm{fl}(s b^{k-1}_{d-i-k,i-j+1,j}) \right. \\
&\qquad\qquad \left. + \mathrm{fl}(t b^{k-1}_{d-i-k,i-j,j+1}))(1+\delta_2)\right](1+\delta_0) \\
&\quad = \left[(r\hat{b}^k_{d-i-k+1,i-j,j})(1+\delta_1) + ((s\hat{b}^{k-1}_{d-i-k,i-j+1,j})(1+\delta_3) \right. \\
&\qquad\qquad \left. + (t\hat{b}^{k-1}_{d-i-k,i-j,j+1})(1+\delta_4))(1+\delta_2)\right](1+\delta_0),
\end{aligned}$$

where $|\delta_i|$, $i = 0, \ldots, 4$, are numbers less than or equal to the unit roundoff u. Then by (4) we can write

$$\begin{aligned}
\hat{b}^k_{d-i-k,i-j,j} = (r\hat{b}^k_{d-i-k+1,i-j,j})(1+\theta_2) + (s\hat{b}^{k-1}_{d-i-k,i-j+1,j})(1+\theta_3) \\
+ (t\hat{b}^{k-1}_{d-i-k,i-j,j+1})(1+\theta_3).
\end{aligned}$$

Iterating the previous argument for $k = d, d-1, \ldots, 1$ and comparing the final expression with (10), we deduce (12) and (13). □

By applying the previous result together with formula (7), which relates backward and forward error, we can derive the following corollary on the forward error of the de Casteljau algorithm.

Corollary 3. *Suppose the hypotheses of Theorem 2 hold. Then the computed value* $\hat{p}(r,s,t) = \mathrm{fl}(p(r,s,t))$ *given by the de Casteljau algorithm satisfies*

$$|p(r,s,t) - \hat{p}(r,s,t)| \leq \gamma_{3d} C_B(p(r,s,t)), \tag{14}$$

where B is the Bernstein basis.

A modified Bernstein-Bézier representation of the polynomial p of (10) was considered in [8],

$$p(r,s,t) = \sum_{i=0}^{d} \sum_{j=0}^{i} c_{d-i,i-j,j} r^{d-i} s^{i-j} t^j, \tag{15}$$

where the new coefficients are related with those of (10) by

$$c_{d-i,i-j,j} = \frac{d!}{(d-i)!(i-j)!j!} b_{d-i,i-j,j}, \qquad 0 \leq j, i \leq d.$$

The following algorithm (which will be called the VS algorithm) was introduced in [8] by Schumaker and Volk to evaluate a polynomial p in the modified Bernstein-Bézier form (15). In contrast with de Casteljau algorithm, this algorithm is expressed in terms of nested multiplications. This version of the algorithm will use the quotients r/t or s/t, assuming that t is bigger than

r and s, in order to avoid divisions by zero or by near-zero values. If r or s is the biggest, it is recommended to adapt the algorithm appropriately.

VS algorithm

$A := c_{d,0,0}$
for $i = 1$ **to** d
  $B := c_{d-i,i,0}$
  for $j = 1$ **to** i
    $B := B \cdot (s/t) + c_{d-i,i-j,j}$
  end j
  $A := A \cdot (r/t) + B$
end i
$p(r,s,t) = A \cdot t^d$.

The previous algorithm requires $(d^2 + 5d)/2$ multiplications and two divisions, and so is significantly faster than the de Casteljau algorithm. A backward error analysis of the VS algorithm is performed in the following result.

Theorem 4. *Consider the VS algorithm in (15), and assume that* $4du < 1$. *Then the computed value* $\hat{p}(x) = \mathrm{fl}(p(x))$ *satisfies*

$$\hat{p}(x) = \sum_{i=0}^{d} \sum_{j=0}^{i} \bar{c}_{d-i,i-j,j} r^{d-i} s^{i-j} t^j, \tag{16}$$

where

$$\frac{|\bar{c}_{i,j,k} - c_{i,j,k}|}{|c_{i,j,k}|} \leq \gamma_{4d}. \tag{17}$$

Proof: The VS algorithm consists of a Horner type algorithm which calculates $\frac{p(r,s,t)}{t^d}$ and a last step which multiplies by t^d. From (15) we can write

$$\frac{p(r,s,t)}{t^d} = \sum_{i=0}^{d} \sum_{j=0}^{i} f_{d-i,i-j,j} r^{d-i} s^{i-j} t^j,$$

with $f_{d-i,i-j,j} = \frac{c_{d-i,i-j,j}}{t^d}$. Since by (1) $\mathrm{fl}(r/t) = (r/t)(1+\theta_1)$ and $\mathrm{fl}(s/t) = (s/t)(1+\theta_1)$, we can apply Theorem 3.1 of [7] to the Horner type part of the VS-algorithm, and get

$$\mathrm{fl}\left(\frac{p(r,s,t)}{t^d}\right) = \sum_{i=0}^{d} \sum_{j=0}^{i} \bar{f}_{d-i,i-j,j} r^{d-i} s^{i-j} t^j,$$

where

$$\bar{f}_{d-i,i-j,j} = f_{d-i,i-j,j}(1 + \theta_{3d+1}). \tag{18}$$

Finally, in the last step we have to multiply by t^d, and then by applying $d-1$ times (1), we can obtain (16) with $\bar{c}_{d-i,i-j,j} = \mathrm{fl}(t^d \bar{f}_{d-i,i-j,j}) = t^d \bar{f}_{d-i,i-j,j}(1+\theta_{d-1})$. Thus by (18) and (3)

$$\begin{aligned}\bar{c}_{d-i,i-j,j} &= t^d f_{d-i,i-j,j}(1+\theta_{3d+1})(1+\theta_{d-1}) = t^d f_{d-i,i-j,j}(1+\theta_{4d}) \\ &= c_{d-i,i-j,j}(1+\theta_{4d}). \quad \square\end{aligned}$$

As a consequence of Theorem 4, and applying again formula (7), we can perform a forward error analysis of the VS algorithm:

Corollary 5. *Under the assumptions of Theorem 4, the computed value* $\hat{p}(r,s,t) = \mathrm{fl}(p(r,s,t))$ *of the VS algorithm satisfies*

$$|p(r,s,t) - \hat{p}(r,s,t)| \leq \gamma_{4d} C_{\bar{B}}(p(r,s,t)), \tag{19}$$

where $\bar{B}$ *is the basis used in the modified Bernstein-Bézier representation.*

Although in Computer Aided Geometric Design, a bivariate polynomial is usually stored in its Bernstein Bézier form (10) (very close to the modified Bernstein-Bézier representation (15)) we shall compare our error bounds with those obtained by evaluating the polynomial in Taylor form by the bivariate Horner algorithm. Given the triangle T and the polynomial p of total degree d defined on T, let $u = x - x_1$, $v = y - y_1$, where (x_1, y_1) are the cartesian coordinates of a point of T. The Taylor form of p is given by

$$p(u,v) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} a_{i,j} u^i v^j. \tag{20}$$

Bivariate Horner algorithm

$p := a_{0,d}$
for $i = 1$ **to** d
  $A := a_{i,d-i}$
  for $j = 1$ **to** i
    $A := A \cdot u + a_{i-j,d-i}$
  end j
end i
$p = A \cdot v + A.$

We observe that the bivariate Horner algorithm requires $(d^2+3d)/2$ multiplications.

The following result, which is a consequence of Corollary 3.2 of [7] for the backward error bound and of formula (7) of the present paper for the forward error bound, provides backward and forward error bounds of the bivariate Horner algorithm.

Theorem 6. *Consider the bivariate Horner algorithm in (20), and assume that* $(2d+1)u < 1$. *Then the computed value* $\hat{p}(u,v) = \mathrm{fl}(p(u,v))$ *satisfies*

$$\hat{p}(u,v) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} \bar{a}_{i,j} u^i v^j, \tag{21}$$

where

$$\frac{|\bar{a}_{i,j,k} - a_{i,j,k}|}{|a_{i,j,k}|} \leq \gamma_{2d+1}, \tag{22}$$

and satisfies

$$|p(u,v) - \hat{p}(u,v)| \leq \gamma_{2d+1} C_M(p(u,v)). \tag{23}$$

§4. Conclusions

As mentioned, in the context of Computer Aided Geometric Design, polynomials are usually stored in the Bernstein-Bézier form (10), which is used by the de Casteljau algorithm. Let us observe that the coefficients of the modified Bernstein-Bézier form (15) used by the VS algorithm are related with those of (10) by

$$c_{d-i,i-j,j} = \frac{d!}{(d-i)!(i-j)!j!} b_{d-i,i-j,j}, \ 0 \leq j, i \leq d. \tag{24}$$

The conversion from the Bernstein-Bézier form into the modified Bernstein-Bézier form can be done in $(d^2+3d-4)/2$ multiplications. In [8] the algorithm composed of the conversion from the Bernstein-Bézier form into the modified Bernstein-Bézier form followed by the VS algorithm was called the VSC algorithm. The number of multiplications and divisions required by the VSC algorithm is $d^2 + 4d$.

We have seen that the VS algorithm is considerably more efficient than the de Casteljau algorithm. On the other hand, the bivariate Horner algorithm is also very efficient and has the lowest backward error bound, as one can deduce from the results in the previous section. However the behaviour with respect to the forward errors is different as we shall now show.

Given the bivariate Bernstein basis B defined on a triangle (see (11)), since the barycentric coordinates satisfy $r + s + t = 1$ we can change the variables in the form $(r,s,t) = (u,v,1-u-v)$ and obtain the following expression of the functions in B:

$$B_{i,j}^d(u,v) = \binom{d}{i,j} u^i v^j (1-u-v)^{d-i-j}, \quad u,v \in [0,1],$$

where

$$\binom{d}{i,j} := \frac{d!}{(d-i-j)!i!j!}.$$

With the same change of variables, the functions of the basis $\bar{B}$ (used in the modified Bernstein form) can be written as

$$\bar{B}^d_{i,j}(u,v) = u^i v^j (1-u-v)^{d-i-j}, \quad u,v \in [0,1].$$

Thus, these functions are related with those of B by

$$B^d_{i,j}(u,v) = \binom{d}{i,j} \bar{B}^d_{i,j}(u,v).$$

We see that the basis functions of B are obtained from those of $\bar{B}$ by a positive scaling. Then, using the condition number of (5), it is easy to prove that

$$C_{\bar{B}}(p(u,v)) = C_B(p(u,v)), \quad \forall p, \forall u,v \in [0,1]. \tag{25}$$

On the other hand, the Taylor form uses the power basis M formed by the functions $u^i v^j$, $0 \le i \le d$, $0 \le j \le d-i$, $u,v \in [0,1]$. By formula (100) of [2], the functions of the power basis M can be expressed as

$$u^i v^j = \sum_{k=i}^{n-j} \sum_{l=j}^{n-k} \frac{\binom{k}{i}\binom{l}{j}}{\binom{d}{i,j}} B^d_{kl}(u,v).$$

Since the coefficients are positive and the basis functions are nonnegative, we can deduce from Lemma 1 that $C_B(p(u,v)) \le C_M(p(u,v))$. Therefore, by (25), we conclude for every polynomial $p(u,v)$,

$$C_{\bar{B}}(p(u,v)) = C_B(p(u,v)) \le C_M(p(u,v)), \quad u,v \in [0,1]. \tag{26}$$

In consequence, although the bivariate Horner algorithm provided lower backward error bounds than de Casteljau and VS algorithms, formula (26) shows that these algorithms use better conditioned bases, and this fact reduces their corresponding forward error bounds.

In conclusion, the VS algorithm has more advantages than de Casteljau or Horner algorithms in this context due to the following properties. First, it uses a basis very close to the Bernstein basis, which is more appropriate in the field of Computer Aided Geometric Design than the power basis. Secondly, the bases used by de Casteljau algorithm and the VS algorithm are also better conditioned than the power basis in the considered domain, this last property being convenient from the point of view of forward error analysis. Finally, the VS algorithm has a higher efficiency than the de Casteljau algorithm.

Acknowledgments. This research has been partially supported by the spanish Research Grant DGICYT PB96-0730.

References

1. Farouki, R. T. and T. N. T. Goodman, On the optimal stability of Bernstein basis, Math. Comp. **65** (1996), 1553–1566.
2. Farouki, R. T. and V. T. Rajan, On the numerical condition of polynomials in Bernstein form, Comput. Aided Geom. Design **4** (1987), 191–216.
3. Higham, N. J., *Accuracy and stability of numerical algorithms*, SIAM, Philadelphia, 1996.
4. Mainar, E. and J. M. Peña, Error analysis of corner cutting algorithms, Numerical Algorithms, to appear.
5. Olver, F. W. J., Error bounds for polynomial evaluation and complex arithmetic, IMA J. Numer. Anal. **2** (1982), 377–406.
6. Peña, J. M., B-splines and optimal stability, Math. Comp. **66** (1997), 1555–1560.
7. Peña J. M. and T. Sauer, On the multivariate Horner scheme. SIAM J. Numer. Anal., to appear.
8. Schumaker, L. L. and W.Volk, Efficient evaluation of multivariate polynomials, Comput. Aided Geom. Design **3** (1986), 149–154.
9. Stewart, G. W., Error analysis of the algorithm for shifting the zeros of a polynomial by synthetic division, Math. Comp. **25** (1971), 135–139.
10. Tsao, N. K., Error analysis of splitting algorithms for polynomials, Numer. Math. **32** (1979), 409–421.
11. Wilkinson, J. H., *Rounding errors in algebraic processes*, Notes on Applied Science 32, Her Majesty's Stationery Office, London, 1963.
12. Wozniakowski, H., Rounding error analysis for the evaluation of a polynomial and some of its derivatives, SIAM J. Numer. Anal. **11** (1974), 780–787.

J. M. Peña
Departamento de Matemática Aplicada
Universidad de Zaragoza
50006 Zaragoza, Spain
jmpena@posta.unizar.es

A Basis for Homogeneous Polynomial Solutions to Homogeneous Constant Coefficient PDE's: An Algorithmic Approach through Apolarity

Michel Pocchiola and Gert Vegter

Abstract. Some recent methods of Computer Aided Geometric Design are related to the *apolar bilinear form*, an inner product on the space of homogeneous multivariate polynomials of a fixed degree, already known in 19th century invariant theory. A generalized version of this inner product was introduced in [8] to derive in a straightforward way some of the recent results in CAGD. Here we extend this work by applying it to compute solution spaces of homogeneous constant coefficient PDE's.

§1. The Homogeneous Apolar Bilinear Form

1.1. Review

In [8] we introduced a generalization of the *apolar bilinear form* defined on the space of homogeneous polynomials (of a certain degree, and with a fixed number of variables). This bilinear form, used extensively in the symbolic method of the classical theory of invariants, has been revitalized by Rota and his co-workers, cf [2] and [4]. In CAGD, a similar binary form on the space of *univariate* polynomials of a fixed degree has been studied by Goldman [3]. It is related to the blossoming approach introduced by Ramshaw [7].

In this section we review some of the properties of the apolar bilinear form. Then we extend [8] by studying constant coefficient partial differential equations of the form $p(\partial)f = 0$, where p is a fixed multivariate homogeneous polynomial. In particular, we derive an algorithm computing a basis for solution spaces consisting of homogeneous polynomials of a fixed degree.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 325–334.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

1.2 Vector spaces of forms

Let $e_1, \ldots, e_s$ be the standard basis vectors on $\mathbb{R}^s$, and let $x = (x_1, \ldots, x_s)$ be the standard coordinates on $\mathbb{R}^s$. The standard inner product on $\mathbb{R}^s$ is denoted by $(\cdot, \cdot)$, i.e., $(u, v) = u_1 v_1 + \cdots + u_s v_s$, for $u, v \in \mathbb{R}^s$.

A central object in this paper is the space of real homogeneous polynomials of degree n on $\mathbb{R}^s$, denoted by $\mathcal{H}_n(\mathbb{R}^s)$. A polynomial in $\mathcal{H}_n(\mathbb{R}^s)$ is the sum of monomials of the form $c_\alpha x_1^{\alpha_1} \cdots x_s^{\alpha_s}$, where $c_\alpha \in \mathbb{R}$ and $\alpha = (\alpha_1, \ldots, \alpha_s) \in \mathbb{Z}^s_{\geq 0}$ is a multi-index of weight $|\alpha| = \alpha_1 + \cdots + \alpha_s$. For convenience the monomial $x_1^{\alpha_1} \cdots x_s^{\alpha_s}$ is denoted by x^α. Linear homogeneous polynomials on $\mathbb{R}^s$ are of the form $f(x) = (u, x)$, for some $u \in \mathbb{R}^s$. We denote f by $(u, \cdot)$.

For multi-indices $\alpha = (\alpha_1, \ldots, \alpha_s)$ and $\beta = (\beta_1, \ldots, \beta_s)$ in $\mathbb{Z}^s_{\geq 0}$ we define $\alpha \mid \beta$ iff $\alpha_i \leq \beta_i$ for $i = 1, \cdots, s$. The relation $\mid$ is a partial order on $\mathbb{Z}^s_{\geq 0}$. Note that $\alpha \mid \beta$ iff there is a $\lambda \in \mathbb{Z}^s_{\geq 0}$ such that $\beta = \alpha + \lambda$. We shall write $\alpha + \beta$ if it is not the case that $\alpha \mid \beta$. A monomial order is a linear ordering $\leq_{\text{mon}}$ on $\mathbb{Z}^s_{\geq 0}$ such that (i) if $\alpha \leq_{\text{mon}} \beta$ and $\gamma \in \mathbb{Z}^s_{\geq 0}$, then $\alpha + \gamma \leq_{\text{mon}} \beta + \gamma$, and (ii) $\leq_{\text{mon}}$ is a well-ordering on $\mathbb{Z}^s_{\geq 0}$, i.e., every non-empty subset of $\mathbb{Z}^s_{\geq 0}$ has a smallest element with respect to $\leq_{\text{mon}}$. We use the notation $\alpha <_{\text{mon}} \beta$ in case $\alpha \leq_{\text{mon}} \beta$ and $\alpha \neq \beta$. Furthermore, we use the property that $\alpha \leq_{\text{mon}} \beta$ whenever $\alpha \mid \beta$. Well known examples are the graded (reverse) lexicographic orders, defined by $\alpha \leq_{\text{mon}} \beta$ if $|\alpha| < |\beta|$, or $|\alpha| = |\beta|$ and in $\alpha - \beta$ the left-most (right-most) non-zero entry is negative (positive). Monomial orders play a paramount role in algorithms for multivariate polynomials, especially with regard to termination conditions; See e.g. [1].

The set of multi-indices in $\mathbb{Z}^s_{\geq 0}$ of weight n, denoted by $\Gamma_{s,n}$, is a finite set with $\#\Gamma_{s,n} = \binom{n+s-1}{n}$ elements. For $\alpha \in \Gamma_{s,n}$ the factorial function is defined by $\alpha! = \alpha_1! \cdots \alpha_s!$, and the multinomial coefficient $\binom{n}{\alpha}$ is defined by

$$\binom{n}{\alpha} = \frac{n!}{\alpha_1! \cdots \alpha_s!}.$$

With a polynomial $f(x) = \sum_{\alpha \in \Gamma_{s,m}} c_\alpha x^\alpha$, we associate the homogeneous differential operator $f(\partial) = \sum_{\alpha \in \Gamma_{s,m}} c_\alpha \partial^\alpha$, where $\partial^\alpha = \partial_1^{\alpha_1} \cdots \partial_s^{\alpha_s}$. Here $\partial = (\partial_1, \ldots, \partial_s)$, with $\partial_i = \partial / \partial x_i$. The directional derivative $D_u : \mathcal{H}_n(\mathbb{R}^s) \to \mathcal{H}_{n-1}(\mathbb{R}^s)$ with respect to $u \in \mathbb{R}^s$ is the differential operator (u, ∂), i.e., $D_u = u_1 \partial_1 + \cdots + u_s \partial_s$. Note that $\partial_i = (e_i, \partial) = D_{e_i}$. Considering e_i as a multi-index of weight one, we also have $\partial_i = \partial^{e_i}$.

1.3. Apolar pairing

This subsection is concerned with a straightforward generalization of the rather well-known apolar inner product $[f, g] = f(\partial) g$, defined on the space of homogeneous polynomials $\mathcal{H}_n(\mathbb{R}^s)$. The main result concerns a characterization of this inner product in terms of three simple properties that will be the basis for the construction of special bases of $\mathcal{H}_n(\mathbb{R}^s)$ in later sections.

Definition 1. *For fixed integers m and n, with $0 \leq m \leq n$, the* apolar pairing *is the map*

$$[\cdot\,,\cdot]_{m,n} : \mathcal{H}_m(\mathbb{R}^s) \times \mathcal{H}_n(\mathbb{R}^s) \to \mathcal{H}_{n-m}(\mathbb{R}^s),$$

associating to the homogeneous polynomials $f \in \mathcal{H}_m(\mathbb{R}^s)$ and $g \in \mathcal{H}_n(\mathbb{R}^s)$ the homogeneous polynomial $[f\,,g]_{m,n}$ of degree $n-m$, defined by

$$[f\,,g]_{m,n} = \frac{(n-m)!}{n!} f(\partial)g.$$

Note that we have in fact a family of pairings, one for each pair of integers m and n with $0 \leq m \leq n$. In this paper, the term pairing refers to the whole family of bilinear maps. From now on we shall drop the subscripts m and n, since they are implicitly known as the degree of the first and second argument of the pairing operator.

Theorem 2. *The apolar pairing is the unique bilinear pairing with the following properties:*

1) (Apolar pairing with constants). *For $f \in \mathcal{H}_n(\mathbb{R}^s)$:*

$$[1\,,f] = f,$$

where $1 \in \mathcal{H}_0(\mathbb{R}^s)$ is the constant homogeneous polynomial of degree 0.

2) (Apolar pairing with linear forms). *For $f \in \mathcal{H}_n(\mathbb{R}^s)$ and $u \in \mathbb{R}^s$:*

$$[(u,\cdot)\,,f] = \frac{1}{n} D_u f.$$

3) (Transposition of a homogeneous factor). *For $f_1 \in \mathcal{H}_{m_1}(\mathbb{R}^s)$, $f_2 \in \mathcal{H}_{m_2}(\mathbb{R}^s)$, and $g \in \mathcal{H}_n(\mathbb{R}^s)$, with $m_1 + m_2 \leq n$:*

$$[f_1 f_2\,,g] = [f_1\,,[f_2\,,g]].$$

It is obvious that apolar pairing is a bilinear operator, satisfying these properties. For the proof of uniqueness, we refer to [8]. Identifying the space of zero degree polynomials with $\mathbb{R}$, we see that for $n = m$, apolar pairing corresponds to a real bilinear form on the space of homogeneous polynomials of degree m. The next result states that this bilinear form is even an inner product. Again, the proof is contained in [8].

Proposition 3. *The apolar bilinear form $[\cdot\,,\cdot] : \mathcal{H}_m(\mathbb{R}^s) \times \mathcal{H}_m(\mathbb{R}^s) \to \mathbb{R}$ is an inner product on the space of homogeneous polynomials of degree m.*

1.3. Dual bases

First we recall the definition of a dual basis pair with respect to the apolar inner product $[\cdot\,,\cdot]$ on $\mathcal{H}_m(\mathbb{R}^s)$.

Definition 4. *The dual basis of a basis* $\{f_\alpha \mid \alpha \in \Gamma_{s,m}\}$ *of* $\mathcal{H}_m(\mathbb{R}^s)$ *is a collection* $\{g_\alpha \mid \alpha \in \Gamma_{s,m}\}$ *of polynomials in* $\mathcal{H}_m(\mathbb{R}^s)$ *such that, for* $\alpha, \beta \in \Gamma_{s,m}$*:*

$$[f_\alpha\,,\, g_\beta] = \delta_{\alpha,\beta}.$$

It is easy to prove the standard fact from linear algebra that a dual basis is indeed a basis. Given a dual basis pair, a polynomial $f \in \mathcal{H}_m(\mathbb{R}^s)$ can be expressed with respect to either basis in terms of coefficients depending on the other one:

$$f = \sum_{\alpha\in\Gamma_{s,m}} [g_\alpha\,,\, f] f_\alpha = \sum_{\alpha\in\Gamma_{s,m}} [f_\alpha\,,\, f] g_\alpha. \tag{1}$$

Example. (Dual of homogeneous Bernstein-Bézier basis). Let $\{x^1, \ldots, x^s\}$ be a basis of $\mathbb{R}^s$, and denote by $u_1(x), \ldots, u_s(x)$ the coordinates of any $x \in \mathbb{R}^s$ in this basis. The polynomials

$$B_\alpha(x) = \binom{n}{\alpha} u_1(x)^{\alpha_1} \cdots u_s(x)^{\alpha_s},$$

where $\alpha \in \Gamma_{s,n}$, form the homogeneous Bernstein-Bézier basis of $\mathcal{H}_n(\mathbb{R}^s)$ with respect to the basis $\{x^1, \ldots, x^s\}$ of $\mathbb{R}^s$. Its dual basis consists of the polynomials

$$l_\beta(y) = (x^1, y)^{\beta_1} \cdots (x^s, y)^{\beta_s},$$

i.e., $[B_\alpha\,,\, l_\beta] = \delta_{\alpha,\beta}$. For the proof, see [8].

§2. Solving Homogeneous Constant Coefficient PDE's

We now show how dual bases can be used for the efficient computation of a basis for the solution space of a homogeneous partial differential equation with constant coefficients, i.e., the space

$$\{f \in \mathcal{H}_n(\mathbb{R}^s) \mid p(\partial) f = 0\}. \tag{2}$$

Here $p \in \mathcal{H}_m(\mathbb{R}^s)$ is a polynomial, $p \neq 0$, that will be fixed throughout the paper. Furthermore, m and n denote fixed integers such that $0 \leq m \leq n$. Our approach is both an alternative and an algorithmic counterpart of Pedersen's work [5,6]. These papers deal with algebraic properties of the space of solutions. We continue Pedersen's work by presenting an optimal algorithm for the computation of a basis for the solution space. Our techniques are new, since they are based on properties of dual bases, together with some recursive properties of the apolar bilinear form introduced in [8].

2.1. Characterizing a basis for the space of solutions

From now on we consider a family of functions $\{f_\alpha \mid \alpha \in \mathbb{Z}^s_{\geq 0}\}$ such that (i) $f_\alpha \cdot f_\beta = f_{\alpha+\beta}$, and (ii) for all $n \geq 0$, the set $\{f_\alpha \mid \alpha \in \Gamma_{s,n}\}$ is a basis of $\mathcal{H}_n(\mathbb{R}^s)$. The dual basis of the latter set is denoted by $\{g_\alpha \mid \alpha \in \Gamma_{s,n}\}$. An example of such a pair of bases is formed by the Bernstein-Bézier basis, together with the lineal polynomials introduced in the example at the end of the preceding section.

Lemma 5. *For $\alpha, \beta \in \mathbb{Z}^s_{\geq 0}$ with $|\alpha| \leq |\beta|$:*

$$[f_\alpha, g_\beta] = \begin{cases} g_{\beta-\alpha}, & \text{if } \alpha \mid \beta, \\ 0, & \text{if } \alpha \nmid \beta. \end{cases}$$

Proof: Let $m = |\alpha|$ and $n = |\beta|$, and let $f = [f_\alpha, g_\beta] \in \mathcal{H}_{n-m}(\mathbb{R}^s)$. Consider the apolar inner product $[f_\gamma, f]$, for $\gamma \in \Gamma_{s,n-m}$. Since $f_\alpha \cdot f_\gamma = f_{\alpha+\gamma}$, transposition of a factor (See Theorem 2, part 3) f_α yields: $[f_\gamma, f] = [f_{\alpha+\gamma}, g_\beta] = \delta_{\alpha+\gamma,\beta}$. First consider the case $\alpha \nmid \beta$. Then $\alpha + \gamma \neq \beta$, and hence $[f_\gamma, f] = 0$, for all $\gamma \in \Gamma_{s,n-m}$. Since the apolar pairing is an inner product and $\{f_\gamma \mid \gamma \in \Gamma_{s,n-m}\}$ is a basis of $\mathcal{H}_{n-m}(\mathbb{R}^s)$, it follows that $f = 0$ in this case. If $\alpha \mid \beta$ the previous derivation shows that $[f_\gamma, f] = \delta_{\gamma,\beta-\alpha}$, so identity (1) implies $f = \sum_{\gamma \in \Gamma_{s,n-m}} [f, f_\gamma] g_\gamma = g_{\beta-\alpha}$. □

In the following, our fixed polynomial p in (2) is given in the form

$$p = \sum_{\alpha \in \Gamma_{s,m}} c_\alpha f_\alpha, \text{ where } c_\alpha = [p, g_\alpha].$$

The following result characterizing the kernel of a polynomial differential operator is the key ingredient for the algorithm developed in the next section.

With p we associate the linear map $D_p : \mathcal{H}_n(\mathbb{R}^s) \to \mathcal{H}_{n-m}(\mathbb{R}^s)$ defined by $D_p(f) = [p, f]$, and the map $T_p : \mathcal{H}_{n-m}(\mathbb{R}^s) \to \mathcal{H}_n(\mathbb{R}^s)$ is multiplication by p, i.e., $T_p(f) = p \cdot f$. Given an integer k, and a subspace $U \subset \mathcal{H}_k(\mathbb{R}^s)$, we denote by $U^\perp$ the orthogonal complement of U with respect to the apolar inner product $[\cdot, \cdot]$ on $\mathcal{H}_k(\mathbb{R}^s)$.

Proposition 6.
1) *$\mathrm{Ker} D_p = (\mathrm{Im} T_p)^\perp$.*
2) *The map D_p is onto.*

Proof: Theorem 2, part 3, implies that T_p and D_p are adjoint operators, i.e., $[T_p(f), g] = [f, D_p(g)]$, for $f \in \mathcal{H}_{n-m}(\mathbb{R}^s)$ and $g \in \mathcal{H}_n(\mathbb{R}^s)$. The first claim follows from this identity. Now since T_p is injective, the result of the first part implies that $dim\, \mathrm{Ker} D_p = dim\, \mathcal{H}_n(\mathbb{R}^s) - dim\, \mathcal{H}_{n-m}(\mathbb{R}^s)$. Therefore, $dim\, \mathrm{Im} D_p = dim\, \mathcal{H}_{n-m}(\mathbb{R}^s)$, and hence D_p is onto. □

As a special case, consider the polynomial $p = f_{\alpha_0}$ for some $\alpha_0 \in \Gamma_{s,m}$. According to Lemma 5, $\mathrm{Ker} D_p$ contains g_β whenever $\beta \in \Gamma_{s,n}$ such that $\alpha_0 \nmid \beta$. Since $\alpha_0 \mid \beta$ iff β is of the form $\beta = \alpha_0 + \lambda$ for some $\lambda \in \Gamma_{s,n-m}$, it

follows that $\#\{\beta \in \Gamma_{s,n} \mid \alpha_0 + \beta\} = \#\Gamma_{s,n} - \#\Gamma_{s,n-m} = dim\, \mathrm{Ker} D_p$. The last equality follows from Proposition 6, part 1. Therefore, a basis for the solution space $\mathrm{Ker} D_p$ is the collection $\{g_\beta \mid \beta \in \Gamma_{s,n} \text{ and } \alpha_0 + \beta\}$. The following result generalizes this special case.

Theorem 7. (Basis for solution space of PDE). *Let $\leq_{\mathrm{mon}}$ be a monomial order on $\mathbb{Z}^s_{\geq 0}$ and $\alpha_0 \in \Gamma_{s,m}$ be defined by $\alpha_0 = min_{\leq_{\mathrm{mon}}}\{\alpha \in \Gamma_{s,m} \mid [p, g_\alpha] \neq 0\}$. Furthermore, for any $\lambda \in \Gamma_{s,n-m}$, let $p_\lambda \in \mathcal{H}_{n-m}(\mathbb{R}^s)$ be the polynomial defined by $p_\lambda = [p, g_{\alpha_0+\lambda}]$. Then*

1) *The set $\mathcal{P}_{n-m} = \{p_\lambda \mid \lambda \in \Gamma_{s,n-m}\}$ is a basis of $\mathcal{H}_{n-m}(\mathbb{R}^s)$.*
2) *Let $\mathcal{Q}_{n-m} = \{q_\lambda \mid \lambda \in \Gamma_{s,n-m}\}$ be the dual of the basis $\mathcal{P}_{n-m}$ of $\mathcal{H}_{n-m}(\mathbb{R}^s)$, i.e., $[p_\lambda, q_\mu] = \delta_{\lambda,\mu}$. A basis for the solution space $\mathrm{Ker} D_p = \{f \in \mathcal{H}_n(\mathbb{R}^s) \mid p(\partial) f = 0\}$ is the set*

$$\{\overline{g}_\beta \mid \beta \in \Gamma_{s,n} \text{ with } \alpha_0 + \beta\}, \tag{3}$$

where $\overline{g}_\beta \in \mathcal{H}_n(\mathbb{R}^s)$ is defined by

$$\overline{g}_\beta = g_\beta - \sum_{\lambda \in \Gamma_{s,n-m}} [p \cdot q_\lambda, g_\beta]\, g_{\alpha_0+\lambda}.$$

Remark. The first claim of Theorem 7 is not necessarily true for other choices of α_0. Consider e.g. the polynomial $p(x) = 2x_1^2 + 2x_1x_2 + x_2^2$, and let $\alpha_0 = (1,1) \in \Gamma_{2,2}$. Here we take the monomial basis for the space of polynomials of degree n on $\mathbb{R}^2$, i.e., we take $f_\beta(x) = x^\beta$, for $x \in \mathbb{R}^2$ and $|\beta| = \beta_1 + \beta_2 = n$. The dual basis consists of the functions g_β, where $g_\beta(x) = \binom{n}{\beta} x^\beta$. For $\lambda \in \Gamma_{2,2}$ we have $g_{\alpha_0+\lambda}(x) = \binom{n}{\alpha_0+\lambda} x_1^{\lambda_1+1} x_2^{\lambda_2+1}$. Take $q(x) = 2x_1^2 - 2x_1x_2 + x_2^2$, then $p(x) \cdot q(x) = 4x_1^4 + x_2^4$, and hence, for all $\lambda \in \Gamma_{2,2}$:

$$[p_\lambda, q] = [[p, g_{\alpha_0+\lambda}], q] = [p \cdot q, g_{\alpha_0+\lambda}] = 0,$$

yet $q \neq 0$. Hence the functions p_λ, where λ ranges over $\Gamma_{2,2}$, do not constitute a basis for $\mathcal{H}_2(\mathbb{R}^2)$.

To prove Theorem 7 we need the following two lemmas.

Lemma 8. *For $\lambda, \mu \in \Gamma_{s,n-m}$ we have*

$$[p_\lambda, f_\mu] = \begin{cases} c_{\alpha_0+\lambda-\mu}, & \text{if } \mu \mid \alpha_0 + \lambda, \\ 0, & \text{otherwise.} \end{cases}$$

Proof: The proof consists of a straightforward calculation:

$$\begin{aligned} [p_\lambda, f_\mu] = [f_\mu, [p, g_{\alpha_0+\lambda}]] = [p, [f_\mu, g_{\alpha_0+\lambda}]] \\ = \begin{cases} [p, g_{\alpha_0+\lambda-\mu}], & \text{if } \mu \mid \alpha_0 + \lambda, \\ 0, & \text{otherwise.} \end{cases} \end{aligned}$$

The last identity is justified by applying Lemma 5. □

Lemma 9. *Let $\lambda_0 \in \Gamma_{s,n-m}$. For $f \in \mathcal{H}_{n-m}(\mathbb{R}^s)$, the following statements are equivalent:*

1) $[g_\lambda , f] = 0$, *for all* $\lambda \leq_{\text{mon}} \lambda_0$,
2) $[p_\lambda , f] = 0$, *for all* $\lambda \leq_{\text{mon}} \lambda_0$.

Proof: It follows from (1) that

$$[p_\lambda , f] = \sum_{\mu \in \Gamma_{s,n-m}} [p_\lambda , f_\mu][g_\mu , f]. \tag{4}$$

Consider $\mu \in \Gamma_{s,n-m}$ such that $\lambda <_{\text{mon}} \mu$, then $\alpha_0 + \lambda - \mu <_{\text{mon}} \alpha_0$. Therefore, the definition of α_0 implies $c_{\alpha_0+\lambda-\mu} = 0$. In view of Lemma 8, we know that $[p_\lambda , f_\mu]$ is equal either to $c_{\alpha_0+\lambda-\mu}$ or to 0, so in any case we have $[p_\lambda , f_\mu] = 0$. This observation allows us to write (4) as

$$[p_\lambda , f] = c_{\alpha_0}[g_\lambda , f] + \sum_{\mu : \mu <_{\text{mon}} \lambda} [p_\lambda , f_\mu][g_\mu , f]. \tag{5}$$

This identity shows that the first statement implies the second one. So assume that statement 2 holds. We may assume that $f \neq 0$, otherwise there is nothing to prove. Let λ_1 be the least multi-index with respect to the monomial order $\leq_{\text{mon}}$ such that $[g_{\lambda_1} , f] \neq 0$. Then (5) implies $[p_{\lambda_1} , f] = c_{\alpha_0}[g_{\lambda_1} , f]$ is nonzero. Hence $\lambda_0 <_{\text{mon}} \lambda_1$. Consequently $[g_\lambda , f] = 0$ for all $\lambda \leq_{\text{mon}} \lambda_0$, which is statement 1. □

Proof of Theorem 7: Let $U \subset \mathcal{H}_{n-m}(\mathbb{R}^s)$ be the space spanned by the p_λ, $\lambda \in \Gamma_{s,n-m}$. Since $\#\Gamma_{s,n-m} = dim\, \mathcal{H}_{n-m}(\mathbb{R}^s)$, it is sufficient to prove that $U = \mathcal{H}_{n-m}(\mathbb{R}^s)$, or, equivalently, that $U^\perp = \{0\}$. Thus, if $f \in U^\perp$, then $[f , p_\lambda] = 0$, for all $\lambda \in \Gamma_{s,n-m}$. According to Lemma 9 this implies $[f , p_\lambda] = 0$, for all $\lambda \in \Gamma_{s,n-m}$, so $f = 0$. This proves 1). Now in view of Proposition 6, the space $\text{Ker}D_p$ is of dimension $\#\Gamma_{s,n} - \#\Gamma_{s,n-m}$, i.e., of dimension $\#\{\beta \in \Gamma_{s,n} \mid \alpha_0 + \beta\}$. On the other hand, it is straightforward to see that the polynomials $\overline{g}_\beta$, $\beta \in \Gamma_{s,n}$ with $\alpha_0 + \beta$, are linearly independent. Therefore, in order to prove that they form a basis of $\text{Ker}D_p$ we just have to prove that they belong to $\text{Ker}D_p$. Taking Proposition 6, part 1, into account, we actually have to check that $[p \cdot q_\mu , \overline{g}_\beta] = 0$, for all $\beta \in \Gamma_{s,n}$ with $\alpha_0 + \beta$. Since

$$\begin{aligned} p \cdot q_\mu &= \sum_{\gamma \in \Gamma_{s,n}} [p \cdot q_\mu , g_\gamma] f_\gamma \\ &= \sum_{\lambda \in \Gamma_{s,n-m}} [[p , g_{\alpha_0+\lambda}] , q_\mu] f_{\alpha_0+\lambda} + \sum_{\substack{\gamma \in \Gamma_{s,n} \\ \alpha_0 + \gamma}} [p \cdot q_\mu , g_\gamma] f_\gamma \\ &= f_{\alpha_0+\mu} + \sum_{\substack{\gamma \in \Gamma_{s,n} \\ \alpha_0 + \gamma}} [p \cdot q_\mu , g_\gamma] f_\gamma, \end{aligned}$$

it follows that $[p \cdot q_\mu , \overline{g}_\beta] = 0$, for all $\beta \in \Gamma_{s,n}$ with $\alpha_0 + \beta$. □

2.2. Computing a basis for the space of solutions

We now present a simple, efficient algorithm for computing the dual basis $\mathcal{Q}_{n-m}$, as well as an example showing how the algorithm works. Recall that for $\alpha = (\alpha_1, \ldots, \alpha_s) \in \Gamma_{s,m}$, the number c_α is equal to $[p, g_\alpha]$. We extend this definition to $\alpha \in \mathbb{Z}^s$ by putting $c_\alpha = 0$ in case at least one of the entries $\alpha_1, \ldots, \alpha_s$ is negative.

Corollary 10.

1) *The dual basis* $\mathcal{Q}_{n-m} = \{q_\mu \mid \mu \in \Gamma_{s,n-m}\}$ *of* $\mathcal{P}_{n-m}$ *is defined recursively by*

$$q_\mu = \frac{1}{c_{\alpha_0}}\Big(f_\mu - \sum_{\substack{\nu \in \Gamma_{s,n-m} \\ \mu <_{\text{mon}} \nu}} c_{\alpha_0+\nu-\mu} q_\nu\Big).$$

2) *For* $\beta \in \Gamma_{s,n}$, *with* $\alpha_0 + \beta$, *the basis function* $\overline{g}_\beta \in \mathcal{H}_n(\mathbb{R}^s)$, *is of the form*

$$\overline{g}_\beta = g_\beta - \sum_{\mu \in \Gamma_{s,n-m}} a_{\mu\beta}\, g_{\alpha_0+\mu}.$$

where the coefficients $a_{\mu\beta}$ *are defined recursively, for* $\mu \in \Gamma_{s,n-m}$, *by*

$$a_{\mu\beta} = \frac{1}{c_{\alpha_0}}\Big(c_{\beta-\mu} - \sum_{\substack{\nu \in \Gamma_{s,n-m} \\ \mu <_{\text{mon}} \nu}} c_{\alpha_0+\nu-\mu} a_{\nu\beta}\Big).$$

Proof: Recall that we are looking for a set of functions $\mathcal{Q}_{n-m} = \{q_\mu \mid \mu \in \Gamma_{s,n-m}\}$, such that $[p_\nu, q_\mu] = \delta_{\nu,\mu}$. In particular, according to Lemma 9 the functions q_μ satisfy $[g_\nu, q_\mu] = 0$, for $\nu <_{\text{mon}} \mu$. Therefore, $q_\mu \in \text{Span}\{f_\nu \mid \nu \in \Gamma_{s,n-m}$ and $\mu \leq_{\text{mon}} \nu\}$, or, equivalently:

$$q_\mu \in \text{Span}(\{q_\nu \mid \nu \in \Gamma_{s,n-m} \text{ and } \mu <_{\text{mon}} \nu\} \cup \{f_\mu\}). \tag{6}$$

Assume we have determined q_ν for $\nu \in \Gamma_{s,n-m}$ with $\mu <_{\text{mon}} \nu$. To compute q_μ satisfying (6), we have to determine constants $d_{\mu\nu}$, for $\mu, \nu \in \Gamma_{s,n-m}$ with $\mu \leq_{\text{mon}} \nu$, such that

$$q_\mu = d_{\mu\mu} f_\mu + \sum_{\substack{\lambda \in \Gamma_{s,n-m} \\ \mu <_{\text{mon}} \lambda}} d_{\mu\lambda} q_\lambda.$$

Since $\mathcal{P}_{n-m}$ and $\mathcal{Q}_{n-m}$ are dual bases, the constants $d_{\mu\nu}$ are uniquely determined by the condition $[p_\nu, q_\mu] = \delta_{\nu,\mu}$. Combining the last two identities we see that

$$[p_\nu, q_\mu] = d_{\mu\mu}[p_\nu, f_\mu] + \sum_{\substack{\lambda \in \Gamma_{s,n-m} \\ \mu <_{\text{mon}} \lambda}} d_{\mu\lambda} \delta_{\lambda,\nu}.$$

From this identity, which holds for all $\nu \in \Gamma_{s,n-m}$ with $\mu \leq_{\text{mon}} \nu$, we derive

$$d_{\mu\mu} = \frac{1}{c_{\alpha_0}},$$

$$d_{\mu\nu} = -\frac{c_{\alpha_0-\mu+\nu}}{c_{\alpha_0}}, \text{ for } \mu <_{\text{mon}} \nu,$$

which proves the first part. Now put $a_{\mu\beta} = [p \cdot q_\mu , g_\beta]$ in (3). Then, according to part 1,

$$a_{\mu\beta} = \frac{1}{c_{\alpha_0}}\Big([p \cdot f_\mu , g_\beta] - \sum_{\substack{\nu \in \Gamma_{s,n-m} \\ \mu <_{\text{mon}} \nu}} c_{\alpha_0+\nu-\mu}[p \cdot q_\nu , g_\beta]\Big).$$

Since $[p \cdot f_\mu , g_\beta] = [p, [f_\mu , g_\beta]] = [p, g_{\beta-\mu}]$, the proof is complete. □

The algorithm for computing a basis for the solution space of the partial differential equation $p(\partial)f = 0$ is now simple:

Algorithm (for computing a basis for $\mathrm{Ker}D_p$).

forall $\mu \in \Gamma_{s,n-m}$ in decreasing $<_{\text{mon}}$-order **do**
  forall $\beta \in \Gamma_{s,n}$ with $\alpha_0 + \beta$ **do**

$$a_{\mu\beta} \leftarrow \frac{1}{c_{\alpha_0}}\Big(c_{\beta-\mu} - \sum_{\substack{\nu \in \Gamma_{s,n-m} \\ \mu <_{\text{mon}} \nu}} c_{\alpha_0+\nu-\mu} a_{\nu\beta}\Big).$$

Example. Consider on $\mathbb{R}^3$ the homogeneous constant coefficient PDE

$$\frac{\partial^2 f}{\partial x_1 \partial x_2} - \frac{\partial^2 f}{\partial x_2^2} + \frac{\partial^2 f}{\partial x_2 \partial x_3} = 0,$$

corresponding to the homogeneous polynomial $p(x) = x_1x_2 - x_2^2 + x_2x_3$. In particular, the setting of this example corresponds to $s = 3$ and $m = 2$. We determine a basis for the solution space in $\mathcal{H}_3(\mathbb{R}^3)$, i.e., we take $n = 3$. To this end, consider the graded reverse lexicographical order on $\mathbb{Z}^3_{\geq 0}$. Let $f_\beta(x) = x^\beta$, and let $g_\beta(x) = \binom{n}{\beta} x^\beta$, where $n = |\beta|$. In this example we denote functions indexed by $\alpha = (i, j, k) \in \mathbb{Z}^3_{\geq 0}$, like f_α, by f_{ijk}.

The sets $\{f_\beta \mid \beta \in \Gamma_{s,n}\}$ and $\{g_\beta \mid \beta \in \Gamma_{s,n}\}$ are dual bases, and moreover $f_\alpha \cdot f_\beta = f_{\alpha+\beta}$, so the conditions for applying Theorem 7 and the algorithm from this section are satisfied. In the notation of Theorem 7, we have $\alpha_0 = (1, 1, 0)$. Note that $p = f_{110} - f_{020} + f_{011}$, so $c_{110} = 1$, $c_{020} = -1$ and $c_{011} = 1$, whereas all other coefficients c_{ijk}, with $i + j + k = 2$, are zero. Now the coefficients $a_{\mu\beta}$ are computed according to the algorithm above, in other words we successively determine the rows in the following table (computing for each row the entries in arbitrary order):

$a_{\mu\beta}$	$\beta = 300$	201	102	030	021	012	003
$\mu = 001$	0	0	0	0	−1	1	0
010	0	0	0	−1	1	0	0
100	0	0	0	−1	2	−1	0

This table corresponds to the following seven basis functions of $\mathrm{Ker}D_p$: g_{300}, g_{201}, g_{102}, $g_{030} + g_{120} + g_{210}$, $g_{021} + g_{111} - g_{120} - 2g_{210}$, $g_{012} - g_{111} + g_{210}$, and g_{003}.

These functions can be turned into monomial form by straightforward substition, yielding the following explicit basis for the solution subspace of $\mathcal{H}_3(\mathbb{R}^3)$: x_1^3, $3x_1^2x_3$, $3x_1x_3^2$, $3x_1^2x_2 + 3x_1x_2^2 + x_2^3$, $-6x_1^2x_2 - 3x_1x_2^2 + 6x_1x_2x_3 + 3x_2^2x_3$, $3x_1^2x_2 - 6x_1x_2x_3 + 3x_2x_3^2$, x_3^3.

Acknowledgments. We are greatful to the anonymous referee for corrections and suggestions for improvement.

References

1. Cox, D., J. Little and D. O'Shea, *Ideals, Varieties and Algorithms*, Springer-Verlag, Berlin, 1992.
2. Ehrenborg, R., and G.-C. Rota, Apolarity and canonical forms for homogeneous polynomials, European J. Combinatorics **14** (1993), 157–191.
3. Goldman, R. N., Dual polynomial bases, J. Approx. Theory **79** (1994), 311–346.
4. Kung, J. P. S., and G.-C. Rota, The invariant theory of binary forms, Bull. Amer. Math. Soc. **10** (1984), 27–85.
5. Pedersen, P. S., A function theory for finding a basis for all polynomial solutions to linear constant coefficient PDE's of homogeneous order, Complex Variables Theory Applications **24** (1993), 79–87.
6. Pedersen, P. S., A basis for polynomial solutions to systems of linear constant coefficient PDE's, Advances in Mathematics **117** (1996), 157–163.
7. Ramshaw, L., Blossoms are polar forms, Comput. Aided Geom. Design **6** (1989), 323–358.
8. Vegter, G., The apolar bilinear form in geometric modeling, Math. Comp. **69** (2000), 691–720.

Michel Pocchiola
Département d'Informatique
Ecole Normale Supérieure
45, rue d'Ulm
75230 Paris Cedex 05, France
Michel.Pocchiola@dmi.ens.fr
http://www.ens.fr/~pocchiol/

Gert Vegter
Department of Mathematics and Computing Science
University of Groningen
P.O. Box 800
9700 AV Groningen, The Netherlands
gert@cs.rug.nl
http://www.cs.rug.nl/~gert/

Triangular G^2-Splines

Hartmut Prautzsch and Georg Umlauf

Abstract. We introduce curvature continuous regular free-form surfaces with triangular control nets. These surfaces are composed of quartic box spline surfaces, and are piecewise polynomial multisided patches of total degree 8 which minimize some energy integral. The Bézier nets can be computed efficiently from the spline control net by some fixed masks, i.e. matrix multiplications.

§1. Introduction

Most methods known for building G^k-free-form surfaces need polynomials of relatively high degree, namely $\mathcal{O}(k^2)$, see for example [2,3]. Only recently in 1995 this high degree was beaten by two methods giving G^k-free-form surfaces of bidegree $2k+2$ with singular [6] and regular [4] parametrizations, respectively. These low degree surfaces can be represented by a control net [4] or a quasi control net [6], and can be designed so as to allow for subdivision.

In this paper we will transfer the method given in [4] to triangular box splines. Here we restrict ourselves to G^2-surfaces which are the most important for practical applications besides G^1-surfaces. Further details and the general case are presented in [5,7].

This paper is organized as follows. In Section 2 we introduce n-sided G^2-patches. These patches are used together with generalized C^2-box spline surfaces to build surfaces of arbitrary topology. How the free parameters in the construction can be used to generate G^2-splines that minimize certain energy functionals and how these G^2-splines can be generated efficiently will be discussed in Section 3.

§2. p-Patches

The simplest C^2-box splines are those over the three-directional grid of total polynomial degree four. In this paper we consider only these box splines. A quartic box spline surfaces has a regular triangular control net and each of its

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 335–342.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

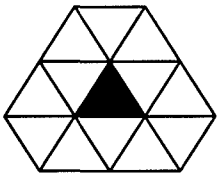


Fig. 1. Schematic illustration of a quartic box spline patch (gray) and its control net.

polynomial patches is determined by 12 vertices (called control points) which are arranged as in Fig. 1.

Furthermore, we can identify in any triangular net regular subnets of the form of Fig. 1. These subnets determine patches forming a generalized box spline surface. A generalized box spline surface has holes corresponding to the irregular vertices in the net. An example is shown in Fig. 2: The control net (left) contains an irregular vertex of valence 8 and the generalized box spline surface (right) has an 8-sided hole.

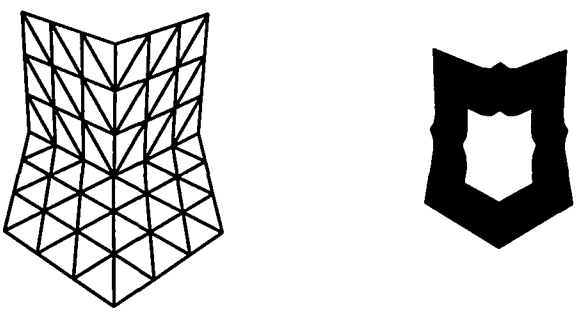


Fig. 2. A triangular net with a vertex of valence 8 (left) and the corresponding generalized quartic box spline surface with an 8-sided hole (right).

If every irregular vertex is surrounded by at least three rings of regular vertices, every irregular vertex corresponds to exactly one hole in the generalized quartic box spline surface. In this case an n-sided hole is surrounded by a complete surface ring consisting of $3n$ patches.

We now describe how to fill such holes with regular G^2-surfaces:

First: for any $n \geq 3, n \neq 6$, we define a special generalized box spline surface that lies in the xy-plane and has the control net shown in Fig. 3 (left) for $n = 5$. Its control points are the points

$$\mathbf{c}_{ijk} := j \begin{bmatrix} c_i \\ s_i \end{bmatrix} + k \begin{bmatrix} c_{i+1} \\ s_{i+1} \end{bmatrix}$$

for $i = 1, \ldots, n$ and $j = 0, \ldots, 3$ and $k = 0, \ldots, 3 - j$, where $c_i = \cos(2\pi i/n)$ and $s_i = \sin(2\pi i/n)$. Thus this surface consists of $3n$ patches, say $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{4n}$, which are shown schematically in Fig. 3 (right).

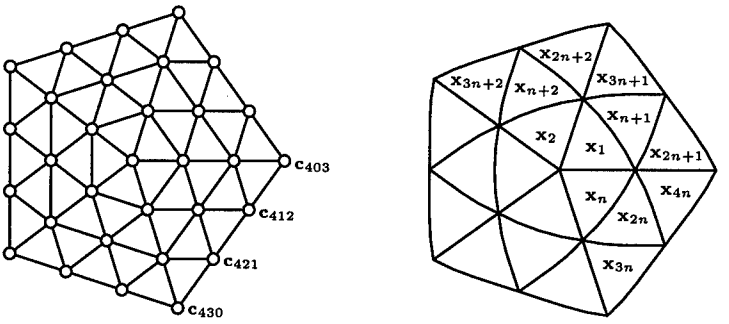


Fig. 3. The control net of $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{4n}$ (left) and the $4n$ planar patches $\mathbf{x}_1, \ldots, \mathbf{x}_{4n}$ (right).

Second: we construct n patches $\mathbf{x}_1, \ldots, \mathbf{x}_n$ filling the hole left by the patches $\mathbf{x}_{n+1}, \ldots, \mathbf{x}_{4n}$, see Fig. 3 (right). Let

$$\mathbf{x}_l(u, v, w) = \sum \mathbf{b}^l_{ijk} B^4_{ijk}(u, v, w)$$

be the quartic Bézier representation of the patch x_l, where u, v, w are barycentric coordinates with respect to some reference triangle, i.e. $u \geq 0, v \geq 0, w \geq 0$ and $u + v + w = 1$. The Bézier points of $\mathbf{x}_l$ are determined such that $\mathbf{x}_l$ has C^2-contact with $\mathbf{x}_{n+l}$. This fixes, say $\mathbf{b}^l_{ijk}$, for $i = 0, 1, 2$. Further, we set $\mathbf{b}^l_{400} = \mathbf{0}$ and $\mathbf{b}^l_{3jk} = \mathbf{b}^l_{2,2j,2k}/2$. Fig. 4 shows these Bézier points for $n = 5$. Note that the scaling differs from Fig. 3.

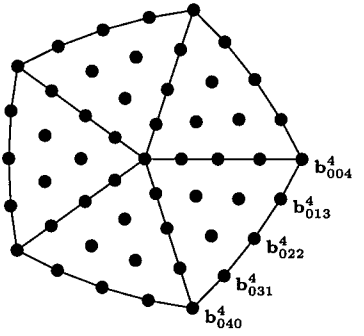


Fig. 4. The Bézier points of $\mathbf{x}_1, \ldots, \mathbf{x}_n$ for $n = 5$.

Lemma 1. *The patches $\mathbf{x}_1, \ldots, \mathbf{x}_{4n}$ are regular and form a surface without self intersections.*

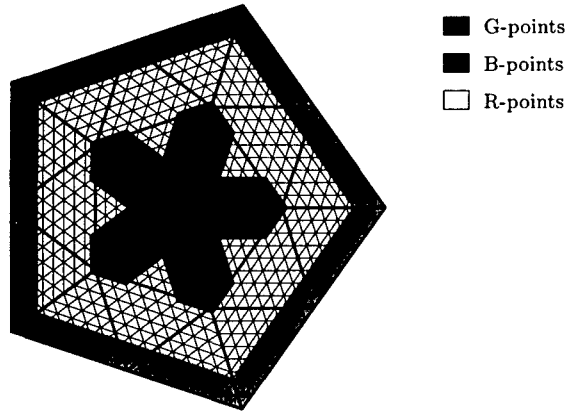


Fig. 5. The Bézier points of a p-patch for $n = 5$.

A proof of this Lemma can be found in [5].

Third: for any polynomial $\mathbf{p} : \mathbb{R}^2 \to \mathbb{R}^3$ we call the union of all patches

$$\mathbf{p}_i(u, v, w) = \mathbf{p}(\mathbf{x}_i(u, v, w)), \quad i = 1, \ldots, 4n,$$

a p-patch. In the sequel we only consider p-patches of degree (4 or) 8 determined by a (linear or) quadratic polynomial $\mathbf{p}$. The Bézier points of such a p-patch are illustrated schematically in Fig. 5 for $n = 5$.

Since for $i = n + 1, \ldots, 2n$ the patch $\mathbf{x}_i$ has C^2-contacts with $\mathbf{x}_{i-n}, \mathbf{x}_{i+n}$ and $\mathbf{x}_{i+2n}$ the patch $\mathbf{p}_i$ also has C^2-contacts with $\mathbf{p}_{i-n}, \mathbf{p}_{i+n}$ and $\mathbf{p}_{i+2n}$. Similarly, $\mathbf{p}_{2n+i}$ and $\mathbf{p}_{3n+i+1}$ have C^2-contact for $i = 1, \ldots, n$, where $\mathbf{p}_{4n+1} = \mathbf{p}_{3n+1}$. Moreover, since a p-patch is part of a polynomial surface each $\mathbf{p}_i, i = 1, \ldots, n$, has G^2-contact with $\mathbf{p}_{i-1}$, where $\mathbf{p}_0 := \mathbf{p}_n$.

The Bézier points that define the G^2-conditions between the patches $\mathbf{p}_1, \ldots, \mathbf{p}_n$ are marked by the underlying dark area in Fig. 5. We call them the G-points of the p-patch. Leaving these points fixed and changing the other Bézier points arbitrarily such that all C^2-joints between adjacent $\mathbf{p}_i$ are preserved, we obtain a modified p-patch. In general, it does not lie on a polynomial surface, but we will still call such a modified p-patch a p-patch.

Theorem 2. *Any n-sided hole of a generalized box spline surface can be filled by a p-patch having a C^2-joint with the generalized box spline surface.*

Proof: The boundary and the cross boundary derivatives up to order two of an n-sided p-patch are determined by $45n$ Bézier points. We call these the B-points of the p-patch. In Fig. 5 they are marked by the grey area.

The B-points can be changed such that the p-patch fits into an n-sided hole of a generalized box spline surface with a C^2-contact. The remaining points without the G-points, here called R-points, can then be adjusted such that any patch $\mathbf{p}_{n+1}, \ldots, \mathbf{p}_{4n}$ of the p-patch has C^2-contact with all its neighbours. Namely, all C^2-conditions involving R-points form a linear system

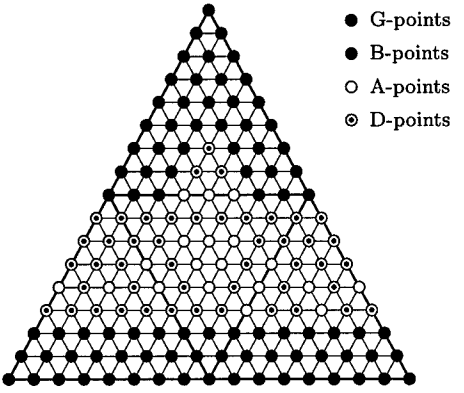


Fig. 6. A possible arrangement of the A-, B-, D- and G-points of $\mathbf{p}_i \cup \mathbf{p}_{i+n} \cup \mathbf{p}_{i+2n} \cup \mathbf{p}_{i+3n}$, $i = 1, \ldots, n$.

for the R-points. The matrix of this system is square if we add enough zero rows. After an appropriate permutation of its columns it is even a block-cyclic matrix. This system has an $18n$ parametric solution. Hence there are $18n$ R-points that can be chosen arbitrarily. We call them A-points. The other R-points are then determined by the A-,B- and G-points via the C^2-constraints. We call these the D-points.

Fig. 6 shows a possible choice for the A- and D-points. Note that this choice is not unique. □

§3. Fair p-Patches

The construction of a p-patch that fills a hole of a generalized box spline surface in Theorem 2 is such that different coordinates do not interfere with each other. So, without loss of generality, we restrict ourselves to scalar valued p-patches in the sequel. Thus a point is no longer a point in $\mathbb{R}^3$, but in $\mathbb{R}^1$.

The G-points of a p-patch are certain Bézier points of a reparametrized quadratic, say

$$q(x,y) = \sum_{i=0}^{2} \sum_{j=0}^{2-i} q_{ij} x^i y^j .$$

Hence the G-points depend linearly on the six coefficients q_{ij}, which we call the Q-points.

Further, as explained in the proof of Theorem 2, the D-points depend linearly on the A-, B- and G-points. Thus, if we consider the B-points fixed, all other Bézier points of the p-patch depend linearly on the six values q_{ij} and the $18n$ A-points.

To obtain good looking surfaces we determine these $6 + 18n$ free parameters such that the p-patch minimizes a quadratic fairness functional. We tried several functionals including the thin plate energy [1]. Judging the visual appearance of the surfaces by their isophotes we got the best results with the

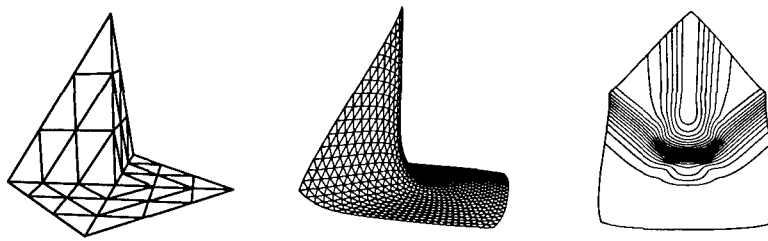


Fig. 7. An initial control net (left), parameter lines of the resulting G^2-surface (middle), top-view of the surface showing isophotes (right).

functional

$$F = \sum_{i=1}^{4n} \int_{\substack{u+v+w=1 \\ u,v,w \geq 0}} \left(\left(\frac{\partial^3}{\partial u^3} \mathbf{p}_i \right)^2 + \left(\frac{\partial^3}{\partial v^3} \mathbf{p}_i \right)^2 + \left(\frac{\partial^3}{\partial w^3} \mathbf{p}_i \right)^2 \right) du\, dv\, dw.$$

The D-points of the p-patch depend linearly on the A-, B- and Q-points. So we can view F as a quadratic functional in the A-, B- and Q-points.

Since F is positive definite, it is minimal for fixed B-points if its derivatives with respect to the A- and Q-points are zero. Differentiating $F = 0$ with respect to the A- and Q-points leads to equations that are linear in the A-, B- and Q-points. Solving for the A- and Q-points shows that the Bézier points of the p-patch minimizing F depend linearly on the B-points. In other words, there is a matrix M_n depending only on F and n such that $M_n \mathbf{b}$ is the vector of all Bézier points if $\mathbf{b}$ is the vector of all B-points.

Fig. 7 shows an example for the G^2-p-patch construction. The initial triangular control net has an irregular vertex of valence 5. The isophotes confirm that the resulting surface is G^2.

Fig. 8 shows a similar example. The control net is the same as in Fig. 7. However, here we used a p-patch consisting of $9n$, $n = 5$, rather that $4n$ patches to fill the n-sided hole of the generalized box spline surfaces.

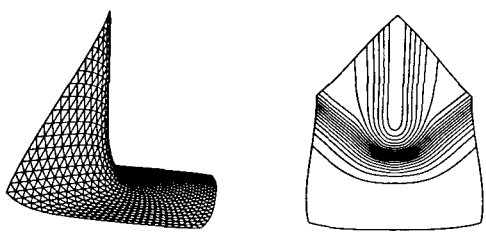


Fig. 8. Parameter lines of the resulting G^2-surface (left), top-view of the surface showing isophotes (right).

A more complex example is shown in Fig. 9. This G^2-surface was computed by the same method as Fig. 7.



Fig. 9. An initial control net (left), the generalized quartic box spline surface with several holes (middle), the resulting surface where every hole is filled with a p-patch (right).

Remark 3. *The matrices M_3, M_4, M_5, M_7, M_8 and M_9 can be found on the website* `http://i33www.ira.uka.de`.

Remark 4. *The construction above can be generalized for generalized box and half box spline surfaces of smoothness order $2k$ and $2k-1$, respectively, see* [5].

Acknowledgments. Supported by IWRMM and DFG grant # PR 565/1-1.

References

1. Florenz, M., Reguläre Flächen auf Dreiecksnetzen beliebiger Topologie, Master's thesis, IBDS, Universität Karlsruhe, 1998.
2. Hahn, J. M., Filling polygonal holes with rectangular patches, in *Theory and Practice in Geometric Modeling*, W. Straßer and H.-P. Seidel (eds.), Springer-Verlag, 1989, 81–91.
3. Mögerle, H., G-Splines höherer Ordnung, PhD thesis, Mathematisches Institut A, Universität Stuttgart, 1992.
4. Prautzsch, H., Freeform splines, Comput. Aided Geom. Design **14** (1997), 201–206.
5. Prautzsch, H., and G. Umlauf, Triangular G^k-splines, IWRMM, Universität Karlsruhe, Preprint 99/8, 1999.
6. Reif, U., TURBS – Topologically unrestricted rational B-splines, Constr. Approx. **14** (1998), 57–78.
7. Umlauf, G., Glatte Freiformflächen und optimierte Unterteilungsalgorithmen, PhD thesis, IBDS, Universität Karlsruhe, 1999. Verlag Shaker, Aachen.

Hartmut Prautzsch
Institute for Operating- and Dialogsystems
Universität Karlsruhe
D-76128 Karlsruhe
Germany
prau@ira.uka.de

Georg Umlauf
Dept. C.I.S.E., CSE Bldg
University of Florida
Gainesville, FL 32611-6120
USA
umlauf@cise.ufl.edu

On Calculating with Lower Order Chebyshev Splines

Mladen Rogina and Tina Bosner

Abstract. We develop a technique to calculate with Chebyshev Splines of orders 3 and 4, based on the known derivative formula for Chebyshev splines and an Oslo type algorithm. We assume that splines in the reduced system are simple enough to calculate. Local bases of Chebyshev splines of order 3 and 4 can thus be evaluated as positive linear combinations of less smooth Chebyshev B-splines. The coefficients in such linear combinations are discrete Chebyshev splines, normalized so as to make a partition of unity. There are a number of interesting special cases, such as Foley's ν-splines, Chebyshev polynomial splines (q-splines), and splines in tension which can be calculated stably by such formulæ.

§1. Introduction and Preliminaries

It is an important fact in the univariate polynomial spline theory that splines can be represented as linear combinations of compactly supported basis functions, which we can calculate in various ways by stable and fast numerical algorithms. This can be extended to some other well-known spaces of splines, such as trigonometric and hyperbolic splines, where nice three-term recurrences of de Boor-Cox type exist; this also applies to a less interesting case of Chebyshev splines with equal weights. These issues have been discussed in [12], where we can also find a negative result concerning the existence of such three-term recurrence relations in general. Polynomials, however, form an algebra, while in other cases multiplicative properties are replaced by other algebraic formulæ, such as addition formulæ for trigonometric/hyperbolic functions or similar identities.

We shall say that an interval $[a,b]$ is measurable with respect to the measure vector $d\boldsymbol{\sigma} := (d\sigma_2, \dots d\sigma_n)^{\mathrm{T}}$ if it is measurable with respect to the positive Lebesgue-Stieltjes measures $d\sigma_i$, $i = 2, \dots n$. Then for $x \in [a,b]$ we can define generalized powers (Chebyshev system) $\{1, u_2, \dots u_n\}$:

$$u_2(x) = \int_a^x d\sigma_2(t_2); \dots; u_n(x) = \int_a^x d\sigma_2(t_2) \dots \int_a^{t_{n-1}} d\sigma_n(t_n). \qquad (1)$$

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 343–352.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

If all of the measures $d\sigma_i$ are dominated by the Lebesgue measure, then they possess densities $\frac{1}{p_i}$, $i = 2, \dots n$; if p_i are smooth, *i.e.*, $\frac{1}{p_i} := \frac{d\sigma_i}{dt} \in C^{n-i+1}$, then $\{1, u_2, \dots u_n\}$ is called an Extended Complete Chebyshev System (ECC-system), referred to as ECT system in [11]). Further, we shall assume that such an integral representation has been found, and the measures for the Chebyshev space determined in such a way that we know the generalized powers explicitly. This may not always be easy, and the choice of measures may not be unique. It is known that Chebyshev B-splines that make a partition of unity exist in this general case, and even that abstract recurrences [5,1] resembling the polynomial ones exist. Almost nothing can be said about their numerical stability, at least not until we employ a special measure vector, whence the abstract construction gets difficult.

Recently, other techniques based on blossoming have been found for the Chebyshev splines [6], which are more promising so far as evaluation and numerical stability is involved. In case of polynomial weights, one obtains Chebyshev polynomial splines [7], though here again the underlying algebraic properties of polynomials are implicitly used.

In Section 2 we give some formulæ for Chebyshev systems of orders 3 and 4 which express locally supported splines (being piecewise in these spaces) as positive linear combinations of less smooth splines in the same space. We will see that the coefficients are related to integrals of splines in the reduced system, which is defined to be a Chebyshev system like (1) corresponding to the reduced measure vector, that is, for each $\delta \subset [a, b]$

$$d\boldsymbol{\sigma}^{(i)}(\delta) := (d\sigma_{i+2}(\delta), \dots, d\sigma_n(\delta))^{\mathrm{T}} \in \mathbb{R}^{n-(i+1)}, \qquad i = 1, \dots n-2.$$

If $\mathcal{S}(n, d\boldsymbol{\sigma}) := \operatorname{span}\{1, u_2, \dots u_n\}$, then the generalized derivatives $L_{j,d\boldsymbol{\sigma}} := D_j \cdots D_1 D_0$, where

$$D_j f(x) := \lim_{\delta \to 0+} \frac{f(x+\delta) - f(x)}{\sigma_{j+1}(x+\delta) - \sigma_{j+1}(x)} \quad j = 1, \dots, n-1, \tag{2}$$

are linear mappings $\mathcal{S}(n, d\boldsymbol{\sigma}) \to \mathcal{S}(n-j, d\boldsymbol{\sigma}^{(j)})$. For a partition $\Delta = \{x_i\}_{i=0}^{k+1}$ of an interval $[a, b]$, and a given multiplicity vector $\boldsymbol{m} = (n_1, \dots, n_k)^T$, $(0 < n_i \leq n)$, we shall denote by $\{t_1 \dots t_{2n+k}\}$ [11] an extended partition in the usual way (see Fig. 1 and Fig. 2):

$$t_1 = \cdots = t_n = x_0,$$
$$t_{n+k+1} = \cdots = t_{2n+k} = x_{k+1},$$

$$t_{n+1} \leq \cdots \leq t_{n+k} = \underbrace{x_1, \dots, x_1}_{n_1}, \dots, \underbrace{x_k, \dots, x_k}_{n_k}.$$

$\mathcal{S}(n, \boldsymbol{m}, d\boldsymbol{\sigma}, \Delta)$ is the spline space spanned by functions being piecewise in $\mathcal{S}(n, d\boldsymbol{\sigma})$ [11]. Chebyshev B-splines in $\mathcal{S}(n, \boldsymbol{m}, d\boldsymbol{\sigma}, \Delta)$ have compact support

$[t_i, t_{i+n}]$, and we shall henceforth assume that these are unique such splines such that

$$\sum_{i=1}^{n+K} T^n_{i,d\boldsymbol{\sigma}^{(j)}}(x) = 1; \quad j = 0, \dots n-2, \tag{3}$$

where $K := \sum_i n_i$. We aim to show that for lower order Chebyshev splines, one can express Chebyshev B-splines as linear combinations with positive coefficients of B-splines with multiple knots. Hopefully, these can be computed efficiently by some interpolating formula or otherwise. In Section 3 we give some examples of how the theory can be used to construct Chebyshev B-splines in some known polynomial cases, like weighted splines and q-splines, but also in the non-trivial case of some other useful Chebyshev splines. To achieve this, we need some technical results. One is the derivative formula, stating that for $x \in [a, b]$, and a multiplicity vector $\boldsymbol{m}$ whose components satisfy $n_i < n-1$ $(i = 1, \dots k)$, the derivative of a T-spline is

$$L_{1,d\boldsymbol{\sigma}} T^n_{i,d\boldsymbol{\sigma}}(x) = \frac{T^{n-1}_{i,d\boldsymbol{\sigma}^{(1)}}(x)}{C_{n-1}(i)} - \frac{T^{n-1}_{i+1,d\boldsymbol{\sigma}^{(1)}}(x)}{C_{n-1}(i+1)}, \tag{4}$$

where

$$C_{n-1}(i) := \int_{t_i}^{t_{i+n-1}} T^{n-1}_{i,d\boldsymbol{\sigma}^{(1)}} d\sigma_2. \tag{5}$$

For continuous $\boldsymbol{\sigma}$, the proof is similar to that in [9]; a somewhat longer proof involving only determinant identities exists, and relies only on the fundamental theorem of integral calculus [11]:

$$f(b) - f(a) = \int_a^b L_{1,d\boldsymbol{\sigma}} f(t) d\sigma_2(t), \tag{6}$$

which holds under very weak hypothesis on the measures.

In certain cases one may construct splines defined on triplets of knots (Lagrangian splines) by a generalized Taylor formula. We give a variant of the Taylor formula, amenable to generalization, that can be useful for lower order Chebyshev systems:

Lemma 1. *If f and $d\boldsymbol{\sigma} = (d\sigma_2, \dots, d\sigma_5)^{\mathrm{T}}$ are such that $L_i f := L_{i,d\boldsymbol{\sigma}} f$ exist and are measurable with respect to $d\sigma_{i+1}, (i = 1, \dots 4)$ on $[a, b]$, then*

$$\begin{aligned} f(x) = & f(a) + L_1 f(a) \int_a^x d\sigma_2(s_2) + L_2 f(a) \int_a^x d\sigma_2(s_2) \int_a^{s_2} d\sigma_3(s_3) \\ & + L_3 f(a) \int_a^x d\sigma_2(s_2) \int_a^{s_2} d\sigma_3(s_3) \int_a^{s_3} d\sigma_4(s_4) \\ & + \int_a^x d\sigma_2(s_2) \int_a^{s_2} d\sigma_3(s_3) \int_a^{s_3} d\sigma_4(s_4) \int_a^{s_4} L_4 f(s_5) d\sigma_5(s_5). \end{aligned}$$

Proof: The hypothesis and (6) enable us to use standard arguments of mathematical analysis in developing a Taylor series expansion

$$\begin{aligned} f(x) =& f(a) + \int_a^x (L_1 f(s_2) - L_1 f(a) + L_1 f(a)) d\sigma_2(s_2) \\ =& f(a) + L_1 f(a) \int_a^x d\sigma_2(s_2) + \text{etc.} \quad \square \end{aligned}$$

Lemma 1 is in fact a way of writing a Taylor expansion for L-splines, see [13], pp. 425-426.

§2. A Knot Insertion Algorithm

We shall henceforth assume that Chebyshev B-splines $T^3_{i,d\boldsymbol{\sigma}^{(1)}}$ can be evaluated at the knots in a numericaly stable way. This is a sound hypothesis, since by (4),

$$T^3_{i,d\boldsymbol{\sigma}}(x) = \frac{1}{C_2(i)} \int_{t_i}^x T^2_{i,d\boldsymbol{\sigma}^{(1)}} d\sigma_2 - \frac{1}{C_2(i+1)} \int_{t_{i+1}}^x T^2_{i+1,d\boldsymbol{\sigma}^{(1)}} d\sigma_2.$$

If we suppose that the multiplicity vector $\boldsymbol{m}$ is such that $T^3_{i,d\boldsymbol{\sigma}} \in C[a,b]$ (that is, $n_i \leq 2$), then we have

$$T^3_{i,d\boldsymbol{\sigma}}(t_{i+1}) = \frac{1}{C_2(i)} \int_{t_i}^{t_{i+1}} T^2_{i,d\boldsymbol{\sigma}^{(1)}} d\sigma_2,$$

and

$$T^3_{i,d\boldsymbol{\sigma}}(t_{i+2}) = 1 - \frac{1}{C_2(i+1)} \int_{t_{i+1}}^{t_{i+2}} T^2_{i+1,d\boldsymbol{\sigma}^{(1)}} d\sigma_2 = \frac{1}{C_2(i+1)} \int_{t_{i+2}}^{t_{i+3}} T^2_{i+1,d\boldsymbol{\sigma}^{(1)}} d\sigma_2.$$

Since the construction of two-interval supported "linear" splines is easy, stable evaluation of $T^3_{i,d\boldsymbol{\sigma}}(t_{i+1})$, $T^3_{i,d\boldsymbol{\sigma}}(t_{i+2})$ amounts to finding an integration formula, preferably of Gaussian type. The important thing is that in this case we do not have any dangerous subtractions potentially leading to large floating point errors, as in (4).

Theorem 2. *Let $T^3_{j,d\boldsymbol{\sigma}^{(1)}} \in \mathcal{S}(3, \boldsymbol{m}, d\boldsymbol{\sigma}^{(1)}, \Delta)$ be a Chebyshev 3rd order B-spline associated with the multiplicity vector $\boldsymbol{m} = (1, \ldots 1)^T$, and let us assume that $\tilde{T}^3_{j,d\boldsymbol{\sigma}^{(1)}} \in \mathcal{S}(3, \tilde{\boldsymbol{m}}, d\boldsymbol{\sigma}^{(1)}), \Delta)$ are Chebyshev B-splines associated with the multiplicity vector $\tilde{\boldsymbol{m}} = (2, \ldots 2)^T$ (Fig. 1) on the same partition. If $\{t_1, \ldots t_{k+6}\}$ and $\{\tilde{t}_1, \ldots \tilde{t}_{2k+6}\}$ are the associated extended partitions, and r an index such that $t_j = \tilde{t}_r < \tilde{t}_{r+1}$, then for $j = 1, \ldots k+3$,*

$$T^3_{j,d\boldsymbol{\sigma}^{(1)}} = T^3_{j,d\boldsymbol{\sigma}^{(1)}}(t_{j+1}) \tilde{T}^3_{r,d\boldsymbol{\sigma}^{(1)}} + \tilde{T}^3_{r+1,d\boldsymbol{\sigma}^{(1)}} + T^3_{j,d\boldsymbol{\sigma}^{(1)}}(t_{j+2}) \tilde{T}^3_{r+2,d\boldsymbol{\sigma}^{(1)}}.$$

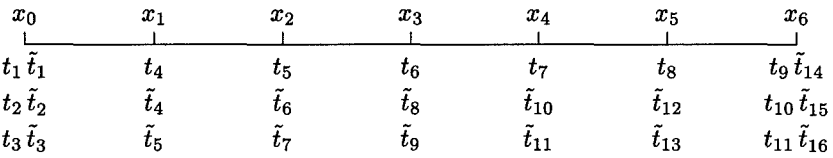


Fig. 1. Double knots.

Proof: Let $T_j^3 := T_{j,d\boldsymbol{\sigma}^{(1)}}^3$. Since $T_j^3(t_j) = T_j^3(t_{j+3}) = 0$ and the same holds for the first generalized derivatives, we conclude that two out of three coefficients representing T_j^3 on each interval of its support are zero. For $x \in (t_j, t_{j+1})$ we have that $T_j^3(t_{j+1}) = \delta_j^3(r)\tilde{T}_r^3(t_{j+1})$. Since (3) applied to $\mathcal{S}(3, \tilde{\boldsymbol{m}}, d\boldsymbol{\sigma}^{(1)}, \Delta)$ implies that $\tilde{T}_r^3(t_{j+1}) = 1$, it remains to show that the middle coefficient equals 1. For $x \in (t_j, t_{j+1})$, property (3) again leads to

$$T_{j-1}^3(x) + T_j^3(x) + T_{j+1}^3(x) = 1.$$

We expand T_i^3, $i \in \{j-1, j, j+1\}$ in $\mathcal{S}(3, \tilde{\boldsymbol{m}}, d\boldsymbol{\sigma}^{(1)}, \Delta)$, and rearrange the terms to obtain

$$\begin{aligned}\tilde{T}_r^3(x)[T_{j-1}^3(t_{j+1}) + T_j^3(t_{j+1})] + \delta_j^3(r+1)\tilde{T}_{r+1}^3(x) \\ + \tilde{T}_{r+2}^3(x)[T_j^3(t_{j+2}) + T_{j+2}^3(t_{j+2})] = 1.\end{aligned}$$

The expressions in square brackets are equal to 1 by (3) applied at the knots t_{j+1}, t_{j+2}. But the partition of unity property must also hold for Chebyshev B-splines in $\mathcal{S}(3, \tilde{\boldsymbol{m}}, d\boldsymbol{\sigma}^{(1)}, \Delta)$, and, since the expansion of unity in this space must be unique, we have $\delta_j^3(r+1) = 1$. □

The more important "cubic" version follows from Theorem 2 and the derivative formula (4):

Theorem 3. *Let $T_{j,d\boldsymbol{\sigma}}^4 \in \mathcal{S}(4, \boldsymbol{m}, d\boldsymbol{\sigma}, \Delta)$, $\tilde{T}_{j,d\boldsymbol{\sigma}}^4 \in \mathcal{S}(4, \tilde{\boldsymbol{m}}, d\boldsymbol{\sigma}, \Delta)$, and let $\boldsymbol{m}$, $\tilde{\boldsymbol{m}}$ be multiplicity vectors as in Theorem 2. Then there exist positive $\delta_j^4(i)$, dependent on $d\sigma_3$, such that $T_{j,d\boldsymbol{\sigma}}^4 = \sum_{i=r}^{r+3} \delta_j^4(i)\tilde{T}_{i,d\boldsymbol{\sigma}}^4$, where $r = r_j$ satisfies $t_j = \tilde{t}_{r_j} < \tilde{t}_{r_j+1}$. If the the extended partition is $\{t_1, \ldots t_{k+8}\}$, and $\{\tilde{t}_1, \ldots \tilde{t}_{2k+8}\}$ is the extended partition with double interior knots, then $\delta_j^4(i)$, $i = r, \ldots r+3$ are explicitly determined by*

$$\delta_j^4(r) = \frac{T_{j,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+1})\tilde{C}(r)}{T_{j,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+1})\tilde{C}(r) + \tilde{C}(r+1) + T_{j,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+2})\tilde{C}(r+2)},$$

$$\delta_j^4(r+1) = \frac{T_{j,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+1})\tilde{C}(r) + \tilde{C}(r+1)}{T_{j,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+1})\tilde{C}(r) + \tilde{C}(r+1) + T_{j,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+2})\tilde{C}(r+2)},$$

$$\delta_j^4(r+2) = \frac{T_{j+1,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+3})\tilde{C}(r+4) + \tilde{C}(r+3)}{T_{j+1,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+2})\tilde{C}(r+2) + \tilde{C}(r+3) + T_{j+1,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+3})\tilde{C}(r+4)},$$

$$\delta_j^4(r+3) = \frac{T_{j+1,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+3})\tilde{C}(r+4)}{T_{j+1,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+2})\tilde{C}(r+2) + \tilde{C}(r+3) + T_{j+1,d\boldsymbol{\sigma}^{(1)}}^3(t_{j+3})\tilde{C}(r+4)},$$

where as in (5),

$$\tilde{C}(i) = \int_{\text{support}} \tilde{T}^3_{i,d\boldsymbol{\sigma}^{(1)}} d\sigma_2. \tag{7}$$

Proof: We expand $T^4_{j,d\boldsymbol{\sigma}}$ in terms of less smooth $\tilde{T}^4_{i,d\boldsymbol{\sigma}}$:

$$T^4_{j,d\boldsymbol{\sigma}}(x) = \sum_{i=r-2}^{r+6} \delta^4_j(i) \tilde{T}^4_{i,d\boldsymbol{\sigma}}(x),$$

and apply (4) to obtain

$$\frac{T^3_{j,d\boldsymbol{\sigma}}(x)}{C_3(j)} - \frac{T^3_{j+1,d\boldsymbol{\sigma}}(x)}{C_3(j+1)} = \sum_i \frac{\delta^4_j(i) - \delta^4_j(i-1)}{\tilde{C}_3(i)} \tilde{T}^3_{i,d\boldsymbol{\sigma}^{(1)}}(x).$$

We may then use Theorem 2 to expand $T^3_{j,d\boldsymbol{\sigma}}$, and write a linear system for δ^4_j, which has the above explicit solution. Details are omitted since the construction very much resembles the construction of ν–splines in [10]. □

It is not difficult to prove that δ^4_j are discrete Chebyshev splines, that is, they form a partition of unity, *i.e.*, $\sum_i \delta^4_i(j) = 1$ [10]; this also holds for general order splines.

We also note that coalescence of the knots yields an expression for a complete "Chebyshev cubic" spline on triple partitions.

§3. Examples

We investigate how much of the above theory can be applied to some known spaces of Chebyshev splines, and what special properties must be used to obtain the stability of the algorithm for the evaluation of Chebyshev B-splines.

Remark 4. Minimizing elastic energy of an inhomogenuous rod leads to a minimization problem for the functional

$$\int_a^b (\mathrm{E}(s)\mathrm{I}(s)u''(s))^2 \, ds \to \min,$$

where E is Young's modulus of elasticity, and I is the moment of inertia of its cross section. We think of the rod as being made of pieces of different material (E piecewise constant), or different cross section (I is piecewise constant) on intervals $[x_i, x_{i+1})$ that partition $[a, b)$. In either case, the Euler equation is $L_{4,d\boldsymbol{\sigma}} u := (w\,u'')'' = 0$, where $d\boldsymbol{\sigma} := (dx, d\sigma, dx)^{\mathrm{T}}$), and $d\sigma$ is the measure generated by the piecewise constant density $w := 1/\mathrm{EI}, w|_{[x_i,x_{i+1}]} =: w_i$.

In CAGD such splines are known as ν–splines, introduced by Foley [2]. It follows from the variational formulation that ν–splines possess continuous derivatives, and that jumps in second derivatives must be continuous:

$$w_{i+1}u''(x_i) - w_i u''(x_i) = 0. \tag{8}$$

If the breakpoints for w are points of the partition Δ, it is not difficult to see (2) that (8) is equivalent to the continuity of the second generalized derivatives $L_{2,d\sigma}$ across the knots. We may evaluate $T^4_{i,d\sigma}$ by scalar products of positive, known quantities; for instance it follows (with the knots enumerated as in Theorem 3), that

$$T_2^4 = \frac{\gamma_2^3(4)}{\|\gamma_2^3\|}\tilde{B}_4^4 + \frac{\gamma_2^3(4)+\gamma_2^3(5)}{\|\gamma_2^3\|}\tilde{B}_5^4 + \frac{\gamma_3^3(7)+\gamma_3^3(8)}{\|\gamma_3^3\|}\tilde{B}_6^4 + \frac{\gamma_3^3(8)}{\|\gamma_3^3\|}\tilde{B}_7^4,$$

where

$$\gamma_2^3(4) = \frac{d\sigma(t_2,t_3)}{d\sigma(t_2,t_4)}(t_4-t_2), \quad \gamma_2^3(5) = t_4 - t_3,$$

$$\gamma_2^3(7) = t_5 - t_4, \quad \gamma_2^3(8) = \frac{d\sigma(t_5,t_6)}{d\sigma(t_4,t_6)}(t_6-t_4),$$

$$\|\gamma_l^3\| = \sum_i \gamma_l^3(i) \quad \text{for} \quad l = 2,3.$$

One should note that $\tilde{B}_i^4$ are ordinary polynomial splines (by raising the multiplicity of the knots we avoid the second derivative condition), and thus are readily calculated by the de Boor-Cox recurrence. This is an example of a Chebyshev system which is not an ECC-system, since $d\sigma$ is generated by a non-continuous density.

Remark 5. If $d\sigma$ in Remark 4 is a Lebesgue measure, the standard knot insertion formula for a "homogenuous rod" *i.e.*, cubic splines appears:

$$B_2^4 = \frac{t_3-t_2}{t_5-t_2}\tilde{B}_4^4 + \frac{t_4-t_2}{t_5-t_2}\tilde{B}_5^4 + \frac{t_6-t_4}{t_6-t_3}\tilde{B}_6^4 + \frac{t_6-t_4}{t_6-t_3}\tilde{B}_7^4.$$

Remark 6. In the last polynomial example, we consider the q-splines introduced by Kulkarni and Laurent [3], which are piecewisely linear combinations of the functions in the canonical system $\{1, u_2, u_3, u_4\}$:

$$u_2(x) = \int_a^x dt_2,$$

$$u_3(x) = \int_a^x dt_2 \int_a^{t_2} q(t_3)\, dt_3,$$

$$u_4(x) = \int_a^x dt_2 \int_a^{t_2} q(t_3)\, dt_3 \int_a^{t_3} dt_4,$$

where q is piecewise linear:

$$q(x)|_{[t_i,t_{i+1}]} := \frac{q_{i+1}-q_i}{h_i}(x-t_i) + q_i,$$

$h_i := t_{i+1} - t_i$, and $q_i > 0$. This may be thought of as an elastic rod with elastic properties changing in a reciprocally linear way; the more physically

x_0	x_1	x_2	x_3	x_4	x_5	x_6
$t_1\,\hat{t}_1$	t_5	t_6	t_7	t_8	t_9	$t_{10}\,\hat{t}_{20}$
$t_2\,\hat{t}_2$	$\hat{t}_5$	$\hat{t}_8$	$\hat{t}_{11}$	$\hat{t}_{14}$	$\hat{t}_{17}$	$t_{11}\,\hat{t}_{21}$
$t_3\,\hat{t}_3$	$\hat{t}_6$	$\hat{t}_9$	$\hat{t}_{12}$	$\hat{t}_{15}$	$\hat{t}_{18}$	$t_{12}\,\hat{t}_{22}$
$t_4\,\hat{t}_4$	$\hat{t}_7$	$\hat{t}_{10}$	$\hat{t}_{13}$	$\hat{t}_{16}$	$\hat{t}_{19}$	$t_{13}\,\hat{t}_{23}$

Fig. 2. Triple knots.

sound model in which material properties are changing linearly does not lead to "Chebyshev polynomial splines" [7], but involves logarithmic weights, and does not seem to be very useful.

Some parts of the construction can be made more explicit, *i.e.*, (7) can be integrated to obtain

$$\tilde{C}_3(r-1) = \frac{h_i}{4}\left[\frac{2q_i}{2q_i+q_{i+1}} + \frac{2q_{i+1}}{q_i+2q_{i+1}}\right],$$
$$\tilde{C}_3(r) = \frac{1}{4}\left[\frac{q_i h_i}{q_i+2q_{i+1}} + h_i + h_{i+1} + \frac{q_{i+2}h_{i+1}}{2q_{i+1}+q_{i+2}}\right].$$

We can stably calculate "parabolic" B-splines required by Theorem 2 in two important points via

$$T_i^3(t_{i+1}) = \frac{h_i(q_i+2q_{i+1})}{6C_2(i)}, \quad T_i^3(t_{i+2}) = \frac{h_{i+2}(2q_{i+2}+q_{i+3})}{6C_2(i+1)},$$

where

$$C_2(i) = \int_{t_i}^{t_{i+2}} B_i^2(t)q(t)\,dt = \frac{1}{6}[(q_i+2q_{i+1})h_i + (2q_{i+1}+q_{i+2})h_{i+1}].$$

One can in fact express $\tilde{T}^4_{i,d\sigma}$ in terms of B-splines on a triple net (Bernstein polynomials). The enumeration of the knots is as in Fig. 2, with quadruple knots at both ends and triple knots $\hat{t}_i$ in the interior. If s is an index such that $t_i = \hat{t}_{s-2} = \hat{t}_{s-1} = \hat{t}_s < \hat{t}_{s+1}$, then

$$\tilde{T}^4_{r-1} = \frac{1}{\tilde{C}_3(r-1)}\frac{2q_i}{2q_i+q_{i+1}}\frac{h_i}{4}\hat{B}^5_{s-2} + \frac{1}{\tilde{C}_3(r)}\Big(\frac{h_i+h_{i+1}}{4}$$
$$+ \frac{q_{i+2}}{2q_{i+1}+q_{i+2}}\frac{h_{i+1}}{4}\Big)\hat{B}^5_{s-1} + \frac{1}{\tilde{C}_3(r)}\frac{q_{i+2}}{2q_{i+1}+q_{i+2}}\frac{h_{i+1}}{4}\hat{B}^5_s,$$

and an analogous result holds for $\tilde{T}^4_r$.

Remark 7. Let us now see how the theory may be applied to the "real" Chebyshev case by considering tension splines with uniform tension parameter that are piecewise in the null-space of the differential operator $D^2(D^2-p^2)$,

where $p \geq 0$ is the tension parameter; see [4] for some more recent references. We may factor the differential operator as

$$D(D + p\tanh px)(D - p\tanh px)D = (\frac{1}{\cosh px}D)(\cosh^2 pxD)(\frac{1}{\cosh px}D)D,$$

and identify the measure vector $d\boldsymbol{\sigma} = (ds_2, \cosh ps_3 ds_3, \frac{ds_4}{\cosh^2 ps_4})^T$. Other measure vectors may be used, but this has the advantage that $L_{1,d\boldsymbol{\sigma}} \equiv D$ maps the tension spline space to the space of hyperbolic splines, where nice recurrencies exist. To apply Theorem 3, we need B-splines in the reduced system [14]:

$$T^3_{j,d\boldsymbol{\sigma}^{(1)}} = \mathrm{C}\begin{cases} \frac{\sinh^2 (p/2)(x-t_j)}{\sinh (p/2)(t_{j+2}-t_j)\sinh (p/2)(t_{j+1}-t_j)}, & x \in (t_j, t_{j+1}), \\ \frac{\sinh^2 (p/2)(x-t_j)\sinh^2 (p/2)(t_{j+2}-x)}{\sinh (p/2)(t_{j+2}-t_j)\sinh (p/2)(t_{j+2}-t_{j+1})} + & \\ \frac{\sinh^2 (p/2)(t_{j+3}-x)\sinh^2 (p/2)(x-t_{j+1}x)}{\sinh (p/2)(t_{j+3}-t_{j+1})\sinh (p/2)(t_{j+2}-t_{j+1})}, & x \in (t_{j+1}, t_{j+2}), \\ \frac{\sinh^2 (p/2)(t_{j+3}-x)}{\sinh (p/2)(t_{j+3}-t_{j+1})\sinh (p/2)(t_{j+3}-t_{j+2})}, & x \in (t_{j+2}, t_{j+3}), \end{cases}$$

where the normalisation constant $C := \cosh \frac{p}{2}(t_{j+2} - t_{j+1})$ ensures the partition of unity (3). The problem of how to calculate $\tilde{T}^4_{i,d\boldsymbol{\sigma}}$ remains. If we use techniques like in Remark 6 to express tension splines on triple nets, we finally arrive at the numericaly nasty formula for a T-spline with support $[t_i, t_{i+1}]$, $(h := t_{i+1} - t_i)$, that is $\hat{T}^4_{i,d\boldsymbol{\sigma}} \frac{1}{\sinh \frac{p}{2}}$ is given by

$$\frac{(\sinh ph - ph)(\cosh p(x - t_i) - 1) - (\cosh ph - 1)(\sinh p(x - t_i) - p(x - t_i))}{2(\sinh ph - ph)(ph/2 \cosh ph/2 - \sinh ph/2)}.$$

The above formula is a special case of the integrated version of the derivative formula (4). Taylor expansions may be used to calculate the above expression for a small p (approx. $p < 0.5$ in double precision, IEEE standard), and also an asymptotic formula for the ultimate almost linear spline, ideas similar to the ones used by Rentrop [8]. In the middle range it is best to utilize the generalized Taylor expansion from Lemma 1 in the vicinity of the inflexion point of $\hat{T}^4_{i,d\boldsymbol{\sigma}}$, which is defined as the solution of the equation $L_{3,d\boldsymbol{\sigma}}\hat{T}^4_{i,d\boldsymbol{\sigma}}(a_{\text{inflex}}) = 0$. On the standard interval $[0, 1]$, one obtains $a_{\text{inflex.}} = \frac{1}{2p} \log \frac{\exp(p)-p-1}{p+\exp(-p)-1}$; both limits never approach boundaries.

For $h = 1$ we can thus obtain an absolutely stable formula in the range $0.5 < p < 700$ with very few arithmetic operations. Translation invariance of splines in tension enables calculation on any interval. We also note that in order to have a complete algorithm, one must have a kind of Gaussian integration formula in closed form to calculate the normalisation constants (5).

Acknowledgments. This research was supported by Grant 037011, by the Ministry of Science and Technology of the Republic of Croatia.

References

1. Dyn, N. and A. Ron, Recurrence relations for Tchebycheffian B-splines, J. Analyse Math. **51** (1988), 118–138.
2. Foley, T. A., A shape preserving interpolant with tension controls, Comput. Aided Geom. Design **5** (1988), 105–118.
3. Kulkarni, R. and P. J. Laurent, Q-splines, Numer. Algorithms **1** (1991), 45–73.
4. Kvasov, B. and P. Sattayatham, Generalized tension B-splines, in *Curves and Surfaces in Geometric Design,* A. LeMéhauté, C. Rabut, and L. L. Schumaker (eds), Vanderbilt University Press, Nashville TN, 1997, 247–254.
5. Lyche, T., A recurrence relation for Chebyshevian B-splines, Constr. Approx. **1** (1985), 155–173.
6. Mazure, M.-L., Blossoming: a geometrical approach, Constr. Approx. **15(1)** (1999), 33–68.
7. Mazure, M.-L. and P. J. Laurent, Polynomial Chebyshev splines, Comput. Aided Geom. Design **16** (1999), 317-343.
8. Rentrop, P., An algorithm for the computation of the exponential splines, Numer. Math. **35** (1980), 81–93.
9. Rogina, M., Basis of splines associated with some singular differential operators, BIT **32** (1992), 496–505.
10. Rogina, M., A knot insertion algorithm for weighted cubic splines, in *Curves and Surfaces in Geometric Design,* A. LeMéhauté, C. Rabut, and L. L. Schumaker (eds), Vanderbilt University Press, Nashville TN, 1997, 387–394.
11. Schumaker, L. L., On Tchebychevian spline functions, J. Approx. Theory **18** (1976), 278–303.
12. Schumaker, L. L., On recursions for generalized splines, J. Approx. Theory **36** (1982), 16–31.
13. Schumaker, L. L., *Spline Functions: Basic Theory,* Wiley, New York, 1981.
14. Schumaker, L. L., On hyperbolic splines, J. Approx. Theory **38** (1983), 144–166.

Mladen Rogina and Tina Bosner
Dept. of Mathematics, University of Zagreb
Bijenička 30, 10000 Zagreb
Croatia
rogina@math.hr

A Declarative Modeler for B-Spline Curves

Vincent Rossignol and Marc Daniel

Abstract. Declarative modeling aims at producing scenes or objects from the user's requirements, and be will briefly introduced. We will then present *MDC*, a Declarative Modeler for Curves, and the different ways for describing curves. We mainly focus on our internal model which allows us to simply manipulate B-splines curves preserving their properties.

§1. Introduction

The current geometric modelers make it possible to construct complex shapes. Nevertheless, the designer has to describe the studied objects by means of lists of coordinates, values or geometric primitives. This way of working, called imperative modeling, is often complex and tedious, even if the associated mathematical models are powerful.

Our goal through declarative modeling is to permit the creation of shapes by only providing a set of abstract specifications, generally based on geometric, topological or physical properties. The role of the computer is then to determine and/or explore the universe of shapes corresponding to the given description. This approach assumes that the description is not overconstrained. Moreover, the time used to describe the shape must be less than the time required to define it by manipulating control points. We are more interested with a "draft" than a very accurate result. A first attempt at declarative modeling of a B-spline curve, and preliminary concepts have been described in [1]. It has led to the new approach proposed in this paper. The method used for the initial description of the properties required by the designer is not very important in the current context, but must be as easy as possible. It can be found in [5]. Declarative modeling is made up of 3 stages:

(1) *the description stage*, where the user's description is transformed into an internal description,
(2) *the generation stage*, where the universe of solution(s) is constructed or sampled from the internal representation,
(3) *the presentation stage*, where solution(s) is (are) presented to the user.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 353–362.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

The first stage is very important but will not be detailed here. It will be just introduced in the next section. References can be found in [4]. The generation stage transfers properties of a virtual curve into geometrical properties applied to the data of our mathematical B-spline model. We finally have to manage a set of control points and a set of geometrical constraints. The constraints link control points to the properties. We chose to focus the paper on the presentation of this model and explain how it is set up.

During the presentation stage, the user has the opportunity to browse through different sets of solutions and to select one of them. Moreover, he/her can interactively move the control points. But each control point can only be moved within a restricted region in which the geometrical constraints are checked.

§2. Curve Description

The description stage in our modeler can be done through 2 methods:

• **The natural description** consists of describing the properties of the curves via pseudo-natural language. For instance, a user can enter a description like: "My curve begins at the top bottom of my workspace", "it has a linear part in the middle", "it has two inflection points"... The description will be translated into a semantic graph that represents properties on the curves which is itself translated into our model presented in next section.

• **The visual description** is another way to enter properties on the curve. For handling properties on the curve, the user can visually insert properties with the mouse on the curve. Then, the computer will ask him for other information. For instance, suppose that we have a curve which corresponds to the natural description seen above. If the user wants to insert a cusp before the linear part, he just has to select this part and asks the computer to insert a cusp there. Then the modeler will ask for the right and the left tangencies. In this mode, work is directly achieved on the internal model.

§3. Constrained B-spline Curves

In this section, we will introduce the internal model for representing and manipulating the curves. It must have three properties:

(1) be as near as possible to the B-spline model,

(2) contain the constraints yielding the description,

(3) allow the user to manipulate the curve preserving the properties.

3.1 Preliminary definitions

Definition 1. *Let P be a point of $\mathbb{R}^2$. We can associate in a formal way a* function of constraint *$\overline{F}_P$ with P whose goal is to restrict any part Z of $\mathbb{R}^2$ according to a property $(\overline{F}_P(Z) \subset Z)$.*

Let C be a B-spline curve with $(n+1)$ control points. We actually are interested in functions of constraint for control points P_i. As these functions often have a generic formulation, it is sometimes convenient to replace the notation $\overline{F}_{P_i}$ with $\overline{F}(.,i)$. A function of constraint reduces the region associated to each constrained point:

Definition 2. *A* constrained point *$(\overline{P})$ is a triplet $(P, Z, \overline{\mathcal{F}})$ defined by:*
P, a point of $\mathbb{R}^2$,
Z, a convex subset of $\mathbb{R}^2$,
$\overline{\mathcal{F}}$, a set of functions of constraint applied to P.

We can now consider a B-spline applying this notion of constrained point. The control polygon is no longer a list of points, but a list of constrained points. It is named a constrained control polygon (next definition). So, with these assumptions, the functions of constraint are set up to ensure properties and to simply manipulate them. An example is proposed in the next section.

Definition 3. *Let C be a cubic B-spline. A* constrained control polygon *is a sequence of constrained points*

$$\overline{\Pi} = (\overline{P}_i)_{i\in\{0,1,\ldots,n\}},$$

with

$$\forall i \in \{0,1,\ldots,n\}, \overline{P}_i = (P_i, Z_i, \overline{\mathcal{F}}_i).$$

Definition 4. *Let $\overline{\Pi}$ be a constrained control polygon on a B-spline C. A constrained point $\overline{P}_i$ is called a* valid constrained point *iff*

$$\forall \overline{F} \in \overline{\mathcal{F}}_i, Z_i \subset \overline{F}(Z_i, i), \qquad (i.e. \quad Z_i = \overline{F}(Z_i, i)),$$

and $Z_i \neq \emptyset$ and $P_i \in Z_i$. If all the constrained points of $\overline{\Pi}$ are valid, $\overline{\Pi}$ is also said to be valid.

3.2 An example of functions of constraint

We choose a simple property "*a linear part on the curve*". We consider a cubic B-spline curve with a uniform knot vector. We define 4 functions called: $\overline{L}_{left}$, $\overline{L}_{midG}$, $\overline{L}_{midL}$ and $\overline{L}_{right}$. We apply these functions on 4 consecutive points of a constrained control polygon. $\overline{L}_{left}$ will be defined by

$$\overline{L}_{left}(Z_i, i) = Z_i \cap [P_{i+1}, \overrightarrow{P_{i+2}P_{i+1}}),$$

where $[P_{i+1}, \overrightarrow{P_{i+2}P_{i+1}})$ represents the ray defined by

$$\{P | \overrightarrow{P_{i+1}P} = k.\overrightarrow{P_{i+2}P_{i+1}}, k \in \mathbb{R}^+\}.$$

One can see in Figure 1 that $\overline{L}_{left}$ has been set up to reduce the region associated with a constrained point. In the same way, we can define $\overline{L}_{midG}$

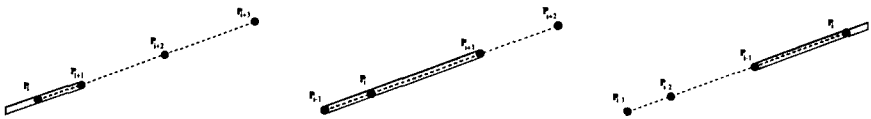


Fig. 1. $\overline{L}_{left}(\mathbb{R}^2, i)$, $\overline{L}_{midG}(\mathbb{R}^2, i)$ and $\overline{L}_{right}(\mathbb{R}^2, i)$.

and $\overline{L}_{midL}$ for the two middle points, and $\overline{L}_{right}$ symmetrically as $\overline{L}_{left}$, see Figure 2:

$$\overline{L}_{midG}(Z_i, i) = Z_i \cap [P_{i-1}P_{i+1}[, \quad \overline{L}_{midL}(Z_i, i) = Z_i \cap]P_{i-1}P_{i+1}],$$

$$\overline{L}_{right}(Z_i, i) = Z_i \cap [P_{i-1}, \overrightarrow{P_{i-2}P_{i-1}}).$$

We can now state the property for the "*linear part*"

Proposition 1 (Linear functions of constraint). *Let C be a cubic B-spline and $\overline{\Pi}$ a constrained control polygon on C. Assume that there exists $i \in \{0, 1, ..., n-3\}$ and*

$$\overline{L}_{left} \in \overline{\mathcal{F}}_i, \quad \overline{L}_{midG} \in \overline{\mathcal{F}}_{i+1}, \quad \overline{L}_{midL} \in \overline{\mathcal{F}}_{i+2}, \quad \overline{L}_{right} \in \overline{\mathcal{F}}_{i+3}.$$

We assume the knot vector to be uniform for the part of the curve associated with $\{P_i, P_{i+1}, P_{i+2}, P_{i+3}\}$. Then if $\overline{\Pi}$ is valid, the curve has a linear part defined by the line segment $[P_{i+1}, P_{i+2}]$.

Proof: The above assumptions imply that for a valid control polygon,

$$Z_i \neq \emptyset, P_i \in [P_{i+1}, \overrightarrow{P_{i+2}P_{i+1}}), \quad Z_{i+1} \neq \emptyset, \quad P_{i+1} \in [P_iP_{i+2}[,$$

$$Z_{i+2} \neq \emptyset, P_{i+2} \in]P_{i+1}P_{i+3}], \quad Z_{i+3} \neq \emptyset, \quad P_{i+3} \in [P_{i+2}, \overrightarrow{P_{i+1}P_{i+2}}),$$

so that points $P_i, P_{i+1}, P_{i+2}, P_{i+3}$ are aligned. We can also notice that this sequence satisfies

$$\overrightarrow{P_iP_{i+1}} = k.\overrightarrow{P_{i+1}P_{i+2}}, \quad \overrightarrow{P_{i+2}P_{i+3}} = k'.\overrightarrow{P_{i+1}P_{i+2}}, \quad (k, k') \in (\mathbb{R}^+)^2$$

Finally, the uniform knot vector yields that the line segment $[P_{i+1}P_{i+2}]$ is a part of C. □

This example is very convenient. But in the same way, we defined functions of constraint for properties like "*cusp*", "*tangencies*" etc.. An introduction to these constraints is available in [6]. This example emphasizes that an organisation for the functions of constraint exists as described in the next section.

3.3 Pieces of control polygon

For defining a linear part on a cubic B-spline, four points are required. Four functions are defined and inserted into the sets of the constrained control points. But these four points are not strongly linked. We introduce a structure linking points: this structure is called Piece of control Polygon (PcP).

Definition 5. *Let $\overline{\Pi}$ be a constrained control polygon on a B-spline curve C. A* Piece of control Polygon *is a triplet (I, m, BP) where*

- *I is the first subscript of the constrained control points associated with the PcP,*
- *m is the number of points associated with the PcP,*
- *BP is a bounding polygon for all the points of the PcP.*

We can now define "*typed PcP*". The type will depend on the property that the PcP handles. For example, a "*linear PcP*" will define a linear part on the curve. In such a case:

- I is the subscript of the first point associated to the linear PcP,
- $m = 4$,
- BP, a bounding polygon for the linear PcP.

Then, if we have

$$\overline{L}_{left} \in \overline{\mathcal{F}}_I, \quad \overline{L}_{midG} \in \overline{\mathcal{F}}_{I+1}, \quad \overline{L}_{midL} \in \overline{\mathcal{F}}_{I+2}, \quad \overline{L}_{right} \in \overline{\mathcal{F}}_{I+3},$$

the curve has a linear part located in bounding polygon BP. As discussed for the linear PcP, different types of PcP can be defined. We have currently implemented, among others, Inflection PcP (for inflection point), Convex PcP, Break point PcP. The curve can now be considered as a sequence of PcP.

Property 1 (Partition of the constrained control polygon). *Consider a constrained control polygon $\overline{\Pi}$ on a B-spline C and a PcP sequence on this polygon. The PcP sequence must partition $\overline{\Pi}$. In other words, each point is included in only one PcP. For a sequence $(PcP_i)_{i\in\{0,1,\ldots,n_{PcP}\}}$, we have $I_0 = 0$, $I_j + m_j = I_{j+1}$, for all $j \in \{0, 1, \ldots, n_{PcP} - 1\}$, and $I_{n_{PcP}} + m_{n_{PcP}} - 1 = n$.*

This property involves two statements. The first is that any point must be in a PcP. A point not pertaining to a PcP does not define any property, and is not required. The second is that a point belongs to only one PcP in order to ensure that no system of constraints on a point will be overconstrained. This approach does not lead to a theoretical minimum number of points defining a B-spline curve. Finally, we can say that the shape of the control polygon on the PcP is based on the shape preversing theorem stated in [2].

3.4 Convexity between two PcP

As the B-spline is defined on $\overline{\Pi}$ and not only on each PcP, B-spline curve segments exist, defined by points belonging to different PcP. These pieces, called "neutral parts", must not introduce unexpected properties. Each neutral part is defined with a control polygon Q_i of four points. In order to avoid unexpected properties in a neutral part, 4 functions of constraints, called joining

functions, are added to preserve the convexity of polygons Q_i. They work similarly to the linear functions seen in Section 3.2. For any PcP, an orientation of the curvature can be associated with each tip of the corresponding curve segment (clockwise or underclockwise). For two consecutive PcP, the orientation of the curvature, called signature (signature *sig* is 1 or -1), must be the same for both neighbouring tips, so that there is no possible inflection on a neutral part.

3.5 Constrained B-spline

Definition 6. *A* constrained B-spline $\overline{C}$ *is defined by*

- k, *its order (currently equal to 4 - i.e.,cubic B-splines)*,
- $n+1$, *the number of its constrained control points*,
- T, *its knot vector*,
- $\overline{\Pi}$, *its constrained control polygon*,
- n_{PcP}, *the number of PcP in* $\overline{\Pi}$,
- PcP, *the sequence of the Pieces of control Polygon*,
- $(sig_i)_{i\in\{2,\ldots,n_{PcP}\}}$, *the signs for the curvature of the neutral parts.*

The first three components correspond to the classical components of a B-spline curve. The knot vector is considered uniform ($t_{n+1} = t_n + 1$) except when a cusp is required on C. In this case, we need to increase the multiplicity of one knot up to 3. For example, if we need to increase the multiplicity at t_p:

$$t_{p-1} = t_p = t_{p+1}, \quad t_{p-2} = t_p - 2, \quad t_{p+2} = t_p + 2.$$

Otherwise, relation $t_{i+1} = t_i + 1$ is preserved.

Definition 7. *Given a constrained B-spline* $\overline{C}$, $\overline{C}$ *is said to be* valid *if its constrained control polygon is valid. This definition is very important. If a constrained B-spline is valid, all the properties imposed through the functions of constraints are checked. We will also see in the next section that a point is allowed to move within its associated region, preserving these properties.*

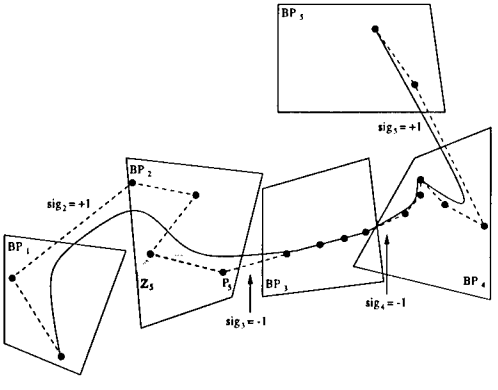


Fig. 2. A complete example of a constrained B-spline.

A complete example of a constrained B-spline curve is illustrated in Figure 2. Three properties are required (an "*inflection point*", a "*linear part*" and a "*break point*"). Region Z_5 is the region where point P_5 can move. All locations in this region preserve the inflection point and the linear part. These locations also ensure that no inflection will be created between the second and the third PcP. Notice that the first and the last PcP are just here to begin and to end curve, and do not have exactly the same behavior as the others.

We defined a function δ which allows us to compute all the regions of the constrained control polygon. For all constrained points, this function initializes the associated region to $\mathbb{R}^2$ and applies all the constrained functions to the region. We can finally state:

Property 2 (Move a point within its region). *Let $\overline{C}$ be a valid B-spline, and i a subscript of a constrained control point ($0 \leq i \leq n$). For all the positions of P_i in Z_i, the properties associated with the functions of constraints are checked.*

This property is one of the most important in the model. When a point is moved within its associated region, a new valid constrained B-spline can be obtained by applying function δ to this new B-spline.

§4. Choosing an Initial Curve

The model we introduced can obtain the different solutions to the designer's problem. Nevertheless, a first curve has to be computed. This section introduces the main steps of this construction algorithm. To solve the problem, we assume that a sequence of PcP is given (this sequence has been constructed during the description stage which is not presented here). Defining the first curve consists in finding a position for all control points so that the application of function δ leads to a valid constrained B-spline. The construction algorithm is divided into 3 stages:

1) Choice of the signature vector,
2) Initialization of the regions,
3) Pick of the control point locations.

4.1 Choice of the signature vector

The signature vector describes the curvature between each PcP. As described in Section 3.4, it is composed of $n_{PcP} - 1$ values in the set $\{+1, -1\}$. All the configurations are not correct. For choosing the values in the vector, we use relations depending on the types of the PcP (see Figure 3). When all the relations between the entries of the vector are defined, an instance of these entries are looked for. The solution is generally not unique. In such a case, different families of solutions have to be investigated (see Figure 4). It may happen that the whole set of relations is inconsistent. This corresponds to an inconsistent description of the curve (for example, "*a closed curve with only one inflection*") and no curve can be computed.

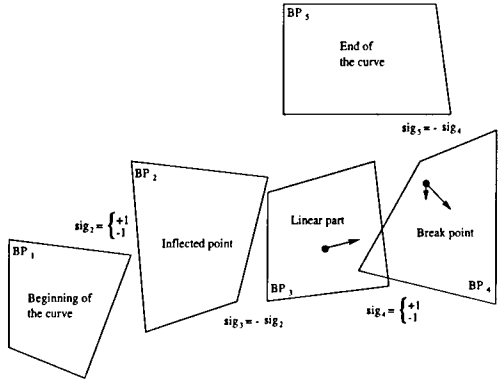


Fig. 3. Relations between signatures.

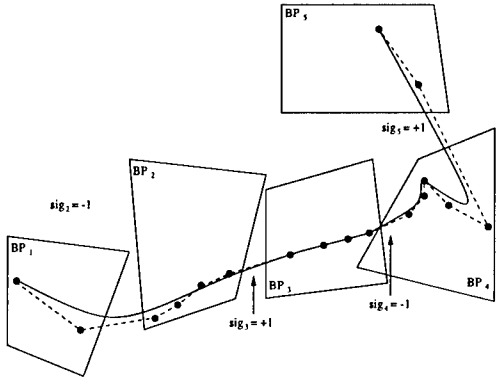


Fig. 4. Another solution than those proposed Figure 2.

4.2 Initialization of the regions

The initialization stage formally sets all the regions to $\mathbb{R}^2$, and reduces them to the bounding polygon of the corresponding PcP. The reduction is then obtained by a geometric construction for each function of constraint, one function at a time. It may happen that an empty region is produced: the description of the curve is inconsistent.

4.3 Choice of the control point location

The method cannot be detailed here. It is divided in two stages:

1) The location of the external (*i.e.*, the first and the last) points of each PcP is determined,
2) The location of the internal points (*i.e.*, all others) are computed.

The first stage is achieved with a specific algorithm. The locations of the first two external points are choosen. An attempt to find the location of

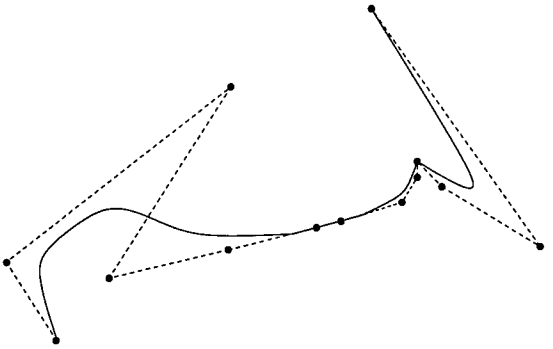


Fig. 5. The curve of Figure 2 defined with less control points.

the next point is made. If no valid location is found, backtracking is started. After a given number of failures, we claim that no solution can be computed, without determining if there is no solution or if we are unable to find it. But generally, a result is produced when it exists. The locations of the internal points are deduced from specific heuristics which never fail.

Once the location of all the control points are computed, applying function δ provides the valid constrained B-spline which can be now manipulated.

§5. Curve Improvement

In order to produce more interesting curves, final improvements have to be applied on the constrained curves. They mainly concern the quality of the control polygon which takes into account the spatial distribution of points, and the reduction of the number of control points.

The quality of the control polygon is defined through a measure of quality (result in interval $[0, 1]$). An increase of the quality is obtained by moving the contrained control points one by one.

As we already mentioned, the number of control points can be too large. Decreasing this number is important while preversing the shape of the curve. General results have been proposed in [3]. As the important properties on the curve and the control points handling these properties are known, our algorithm is easier: first remove non-critical points for the properties, then optimize the distance between the first curve and the reduced one. An example is shown in Figure 5.

§6. Conclusion

MDC validates the approach described in this paper. Improvements of the program are still necessary, but it already provides interesting results. A declarative modeler does not exclude a classical modeler but can provide a way for the user to eliminate the most tedious part of the design process.

The declarative approach has another application: the produced curve can be considered as a classical B-spline. Its properties can be kept so that semantic information is available (which is not so far from form features in CAD). This information would be useful in applying other algorithms afterwards to the curve.

References

1. Daniel, M., Declarative Modeling of fair shapes: An additional approach to curve and surface computations, in *Advanced Course on FAIRSHAPE* (J. Hoschek and P. Kaklis - eds.), B. G. Teubner Stuttgart, 1996, 77–85.
2. Kantorowitz, J. E. and Y. Schechner, Managing the shape of planar splines by their control polygons, Computer-Aided Design **25-6** (1993), 355–364.
3. Schumaker, L. L. and S. S Stanley, Shape-preserving knot removal, Computer Aided Geometric Design **13** (1996), 851–872.
4. Desmontils, E., and J. Y. Martin, Properties Taxonomy in Declarative Modeling, CISST'97, Las Vegas, 130–138.
5. Rossignol, V., and M. Daniel, Le module de description d'un modeleur déclaratif, Research Report IRIN number 177, 1998.
6. Rossignol, V., and M. Daniel, Étude des contraintes dans la phase de génération d'un modeleur déclaratif de courbes, Research Report IRIN number 178, 1998.

Vincent Rossignol
IRIN (Institut de Recherche en Informatique de Nantes)
2, Rue de la Houssinière, BP 92208
44322 Nantes Cedex 3, France
Vincent.Rossignol@irin.univ-nantes.fr

Marc Daniel
XAOlab-ESIL/LIM
Campus de Luminy, case postale 925
13288 Marseille cedex 9, France
Marc.Daniel@esil.univ-mrs.fr

Diffuse Curvature Computation for Surface Recognition

J. M. Savignat, O. Stab, A. Rassineux, and P. Villon

Abstract. Diffuse approximation is a local approximation scheme based on a moving least square fit. Derivatives are estimated by a pseudo-derivation operator which (under certain conditions) converges towards the function derivatives. For this reason, we use it to compute curvature over triangular surfaces as an extention of the fitting algorithm. We also take triangle normals into account, which leads to a high quality curvature estimator. We develop a surface recognition algorithm for triangular surfaces based on this curvature computation on the one hand, and on the topology described by the mesh on the other hand. Its application allows us to treat successfully some real CAD models, implying that diffuse approximation is a powerful tool for surface modelling, and for derivative-based computations.

§1. Diffuse Approximation

We shall focus in this part on the 1D case because the extension to higher dimensions only involves notational difficulties. Given a set of points $(x_i)_{i\in I}$ in $\Omega \subset \mathbb{R}$ an open interval, with measures $(u_i)_{i\in I}$, we build locally an approximation of the underlying function u via an estimation of the Taylor expansion of the function u. It should be noted that for any function $u \in C^{m+1}$, the Taylor expansion of order m exists at each point y,

$$u(x) = \sum_{k=0}^{m} u^{(k)}(y)\frac{(x-y)^k}{k!} + \int_y^x \frac{(t-y)^m}{(m)!}u^{(m+1)}(t)\,dt,$$

and that the polynomial part is an approximation of u near the point y.

The estimate uses some weight functions w_i associated with each point x_i and locally supported around x_i. We define $I(x) = \{i \in I, w_i(x) \neq 0\}$ as the set of indices of data points whose weight function is non-null at x. The computation procedes by minimisation at a point y of the functional

$$\mathcal{E}_y(\{\alpha\}) = \sum_{i\in I} w_i(y)(\bar{u}_y(x_i) - u_i)^2,$$

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 363–370.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

with $\bar{u}_y(x) = \sum_{k=0}^{m} \alpha_k(y) p_y^k(x)$ and $\langle p_y(x) \rangle = \langle 1, (x-y), \ldots, \frac{(x-y)^k}{k!}, \ldots \rangle$. The approximation $\tilde{u}$ of u and its derivatives are the coefficients α_k: $\tilde{u}(y) = \alpha_0(y)$, $\frac{\delta \tilde{u}}{\delta x}(y) = \alpha_1(y)$, ... This approximation method was first proposed in [10], and efficient computation was dicussed in [2].

The diffuse approximation properties depend mainly on the weight functions w_i. Their usual form is $w_i(x) = w_{ref}(\frac{x-x_i}{\rho_i})$, where w_{ref} is a reference bell function with support $(-1, 1)$, and ρ_i is the influence radius of point x_i. We shall suppose that these radii are chosen so that the approximation exists at any point x (i.e. $\forall x \in \Omega, \mathrm{Card}(I(x)) \geq m$). [3] and [4] presented a few techniques to calculate such radii. With these definitions, the main properties of $\tilde{u}$ are the following:

- $\tilde{u}$ has the same smoothness as w_{ref} (e.g. if $w_{ref} \in C^2$, $\tilde{u} \in C^2$).
- The approximation reproduces polynomial functions up to degree m.
- $\tilde{u}$ and the pseudo-derivatives $\frac{\delta^k \tilde{u}}{\delta x^k}$ $(k \leq m)$ converge to u and its derivatives when the number of data points increases (see [16]).
- The diffuse approximation is linear, and the shape functions defined by $\tilde{u}(x) = \sum_{i \in I(x)} N_i(x) u_i$ are local, $\mathrm{supp}(N_i) = \mathrm{supp}(w_i)$.

§2. Hermite Approximation Scheme

We propose to also take differential data into account in the criterion $\mathcal{E}_y$ to build a Hermite approximation scheme. Let $(x_j)_{j \in J}$ denote the set of points at which some differential data $v_j = \mathcal{D}_j(u)(x_j)$ are known. We associate a weight function w_j with each point x_j. The modified criterion is

$$\mathcal{E}_y(\{\alpha\}) = \sum_{i \in I(y)} w_i(y)(\bar{u}_y(x_i) - u_i) + \sum_{j \in J(y)} \lambda_j w_j(y) ||\mathcal{D}_j \bar{u}_y(x_j) - v_j||^2.$$

It is not restrictive to suppose that all the differential operators correspond to the same operator $\mathcal{D} = \{\mathcal{D}^l\}_{l \in [1,n]}$. Then the vector $\{\alpha\}$ is a solution of the system

$$A(y)\{\alpha(y)\} = \{b(y)\},$$

with

$$A(y) = P^\top(y) W(y) P(y) + \lambda \sum_{l=1}^{n} P^{l\top}(y) W^d(y) P^l(y),$$

$$\{b(y)\} = P^\top W(y) U(y) + \lambda \sum_{l=1}^{n} P^{l\top}(y) W^d(y) V^l(y),$$

where $W(y)$ and $W^d(y)$ are the diagonal matrix of weights $w_i(y)$ and $w_j(y)$ respecively, $P(y) = [p_y(x_i)]_{i \in I(y)}$,$P^l(y) = [\mathcal{D}^l(p_y)(x_j)]_{j \in I(y)}$ and $U(y) = \{u_i\}_{i \in I(y)}$, $V(y) = \{v_j\}_{j \in J(y)}$.

The previous properties remain valid for this new formulation. A similar approximation method was proposed in [7] for dealing with boundary conditions in a Galerkin method for partial differential equation.

§3. Curvature Computation

Curvature is mainly used in the treatment of range images (see [1]), and most algorithms were developed for these kind of data. In the paper [9], the authors distinguished four types of algorithms: finite difference methods, the facet model, geometrical methods, and fitting methods. The first two categories only apply to depth maps, whereas the last two are more general.

As the geometrical methods are ad hoc contructions, we shall not examine them in this paper. The facet model described in [5] uses a polynomial fit to compute more accurate finite difference formulas. The same idea was used in [8] for generalized finite differences. Therefore, the last three methods are mainly of the same kind, and a diffuse model gives some theoretical background to them.

Except for geometrical methods, curvature computation at nodes (i.e. data points) is composed of four steps:

1) Extraction of the node neighborhood,
2) Calculation of a coordinate system in which the surface is a Monge patch (i.e. there exists a function φ such that the surface has the form $(x, y, z = \varphi(x, y))$),
3) Evaluation of partial derivatives of the surface at the node,
4) Computation of curvatures.

The finite difference and facet model-based methods precompute some steps to obtain faster estimates. A more extensive bibliography can be found in [9] and [13].

Meshed surface curvature estimation can be done in one of the following three ways:

1) Forget the mesh, and treat a 3D point set,
2) Use the mesh as a purely topological attribute,
3) Use the mesh to interpolate the data.

The paper [15] uses the second strategy: it applies a multiresolution fitting method where the neighborhood of a node is defined through the triangular mesh. " Different layers of connectivities define different levels of neighboring relationships, e.g., the first level neighbors are the point with direct connection with the node, the second level ones have direct connection with the first level neighbors, and so on ". The third solution is difficult because it needs high continuity elements which are difficult to build; but [12] shows a solution based on G^2 continuity which is not robust to element shape [13]. Our method is of the second kind.

§4. Diffuse Curvature Computation

The diffuse curvature computation uses both the point positions and some normals (e.g. triangle normals in this paper). We focus on the computation at the nodes only, which will help in the definition of the parameters of the diffuse algorithm. The polynomial basis is $\langle 1, x, y, x^2, xy, y^2 \rangle$ which allows the

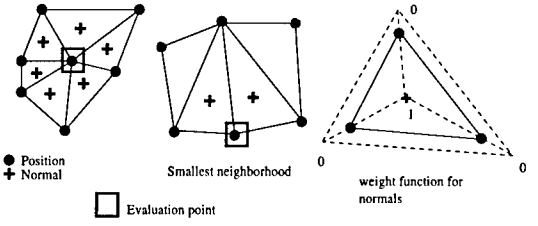


Fig. 1. Neighborhood selection.

evaluation of second-order partial derivatives needed for curvature computation.

We shall follow the four steps described above to compute curvature at a node x_i. Neighborhood extraction is based on the following weight functions: w_i is such that $w_i(x_i) = 1$, $w_i(x_{i'}) = \frac{1}{2}$ if node $x_{i'}$ is connected to x_i and $w_i(x_{i'}) = 0$ otherwise. The weight function of a normal is defined with the dashed triangulation (Figure 1, right). For smoothness reasons, we add the gray nodes to evaluate curvature at nodes on the border with only three edges (Figure 1, center).

We then compute local coordinates using an algorithm based on principal component analysis, and estimate the partial derivatives of the Monge patch (i.e. of function φ) defined by the data points with the pseudo-derivatives of the diffuse approximation at point x_i. Finally, principal curvatures k_1 and k_2 are computed with their associated directions.

A numerical study of the proposed method is given in [13], and shows that it gives at least as good results as the fitting method with smaller dependence neighborhoods, which is an important factor for surface recognition. It shows, moreover, that λ has to be small.

§5. Surface Recognition

Surface recognition is the first step in reconstructing a CAD model from a triangular mesh. We shall suppose in the following that the considered surface satifies the following hypotheses:

H_0: The surface is composed of parts of planes, cylinders, spheres, cones and torii (called patches).

H_1: Each patch contains at least one interior node.

H_2: Patch intersection are contained in the mesh (i.e. they are described by some edges chains).

Under these hypotheses, the above-mentioned diffuse curvature computation always estimates the real surface curvature at nodes interior to a patch (H_1 and H_2), because data are taken from the right surface. This is not the case with usual techniques (mainly the fitting method). This property is essential to proving that the recognition algorithm correctly classifies each node of a surface under the hypotheses H_0, H_1 and H_2 [14].

The recognition algorithm is composed of four steps.

Firstly, an initial classification (based on hypothesis H_0) is proposed with the following rules applied sequentially:

- If $k_1 = k_2 = 0$, the nodes is a PLANE node.
- If $k_2 \neq 0$ and $\frac{k_1}{k_2} = 1$, the node is a SPHERE node.
- If $k_2 \neq 0$ and $\frac{k_1}{k_2} \neq 1$, the node is a TORUS node.

The classification of non-classified nodes uses their comparison with connected nodes:

- If all connected nodes have the same k_1 and associated direction, the node is a CYLINDER node.
- If they have same k_1 and different directions, the node is a TORUS node.
- The node is a CONE node otherwise.

Secondly, we check the consistency of the initial classification with hypotheses H_0, H_1 and H_2. For example, a cone-cylinder intersection node is classified as a TORUS node. The basic idea of this consistency check is that classified connected nodes must form some connected homogeneous sets (as a consequence of H_2). From the study of intersections between the five primitives, it is possible to define three consistency rules (that are shown to be sufficient in [14]):

- For all connected nodes of different kinds: if one of them is a PLANE node unclassify the other one, otherwise if one of them is a CYLINDER node, unclassify the other one. Unclassify both nodes in other cases.
- If two connected nodes are both TORUS nodes with different k_1, unclassify both nodes.
- If two connected nodes are both SPHERE nodes with different k_2, unclassify both nodes.

At this stage, we obtain some germs that are homogeneous connected sets of nodes. We shall grow these germs to classify the whole surface via a marching algorithm.

Thirdly, we consider a classified node $\boldsymbol{n}$ and the patch P to which it belongs. From hypothesis H_2, for any triangle $T = (\boldsymbol{n}, m, p)$ we can claim that

1) T and its edges $\boldsymbol{n}m$ and $\boldsymbol{n}p$ belong to the interior of P,
2) Nodes m, p and edge mp belong to P.

Therefore, nodes m and p are either in the interior of the patch P or on the intersection of P and another patch. This analysis forms the topological operator of the marching algorithm.

The next step is then to check whether nodes m and p are interior nodes or intersection nodes. This decision is based on two rules which use geometrical

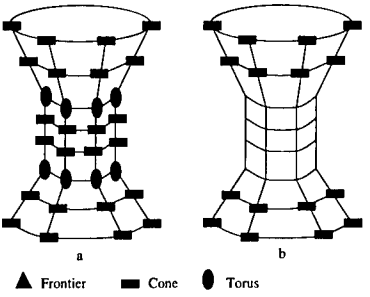


Fig. 2. (a) Initial classification (b) Consistent classification.

information. The first rule concerns connected triangles and their connecting edge.

- If one of the triangles is not classified, do nothing,.
- If both triangles are of the same kind and a vertex of the common edge belongs to an intersection, classify the other vertex as an intersection node.
- If the triangles are of a different kind, the vertices of the common edge are intersection nodes.

The second rule is node-based. It looks at the classification of the connected nodes: If this list is not homogeneous, then the node lies on an intersection. If it is homogeneous, some tests based on the same ideas as the initial classification allow us to check whether the node belongs to a surface or to an intersection of two surfaces of the same kind (this situation may happen after some iterations of the marching algorithm). The iteration of the topological operator and the two classification rules grows the germs of the consistent initial classification.

Figure 2 shows that the initial consistency algorithm may kill all potential germs of some patches. Provided that hypothesis H_1 is valid, some post treatment can be applied to this situation. The basic ideas of these treatments are the same as those being used in the main recognition algorithm. This is the fourth and last step of the classification.

§6. Conclusion

Under the additional hypothesis that curvature computation is exact (H_3), the recognition algorithm is successful i.e. *If the Hypotheses H_0 to H_3 are valid, the recognition algorithm classifies correctly all the nodes of a triangulated surface.* Triangles are classified except for those which are based on three frontier nodes. These triangles can be classified in a subsequent model fitting stage. The third hypothesis is restrictive, but numerical experiments showed that the algorithm is succesfull as soon as the surface satisfies hypotheses H_0, H_1 and H_2.

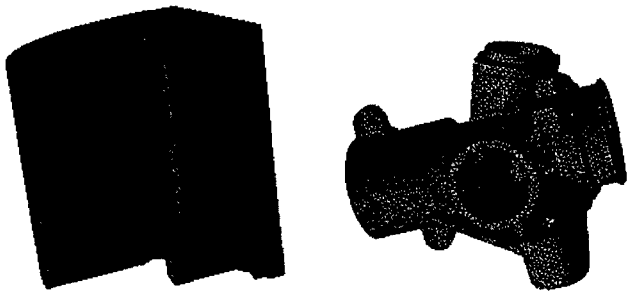


Fig. 3. Two real CAD meshed models.

Table 1 shows the relative number of correctly classified nodes after each step of the algorithm. The first surface satisfies all three hypotheses H_0, H_1 and H_2. The second surface does not satisfy hypotheses H_0 and H_1. As a consequence, some nodes are not classified, and each unclassified node is link to a hypothesis violation.

	example 1	example 2
Initial classification	95 %	99.9 %
Consistency	87 %	80 %
Marching	98 %	88 %
Post treatment	100 %	88 %

Tab. 1. Relative number of classified nodes.

In conclusion, the use of the Hermite approximation scheme we proposed in this paper allows us to build a simple but efficient recognition algorithm. The numerical experiments showed that the Hermite Diffuse Approximation is a powerful tool for surface analysis and partial derivatives estimation. The curvature computation was also used in [11] in a remeshing scheme.

The quality of the curvature estimation on CAD models will help to enlarge the number of surfaces taken into account. Furthermore, [6] showed that Moving Least Square approximation can be applied directly to surface modeling. This approach may be useful for still better curvature estimation.

References

1. Besl, P. J. and R. C. Jain, Three-dimensional object recognition, Computing Surveys **17** (1985), 75–145.
2. Breitkopf, P., G. Touzot, and P. Villon, Explicit form and efficient computation of MLS shape functions and their derivatives, Numerical Method in Engineering, to appear.

3. Duarte, C. A. and J. T. Oden, Hp-clouds, an hp meshless method, Numerical Methods for Partial Differential Equations **16** (1996), 673–705.
4. Guyon, F., Interpolation diffuse. Champs d'atténuation constants, Université de Technologie de Compiègne, 1995.
5. Haralick, R. M. and L. G. Shapiro, *Computer and Robot Vision, Vol 1*, Addison Wesley, New York, 1992.
6. Levin, D., Mesh-independent surface interpolation, Advances in Computational Mathematics, to appear.
7. Liska, T., C. A. M. Duarte, and W. W. Tworzydlo, Hp–meshless cloud method, Computational Methods for Applied Mechanics and Engineering **139** (1996), 3–47.
8. Liska, T. and J. Orkish, The finite difference method at arbitrary irregular grids and its application in applied mechanics, Computers and Structures **11** (1980), 83–95.
9. Mc Ivor, A. M. and P. T. Walkenburg, A comparison of local surface geometry estimation methods, http://www.irl.cri.nz/publications , 1997.
10. McLain, D. H., Drawing contours from arbitrary data points, The Computer Journal **17** (1974), 318–324.
11. Rassineux, A., J. M. Savignat, O. Stab, and P. Villon, Surface remeshing by local Hermite diffuse interpolant, Trends in Unstructured Mesh Generation, USNCCM'99, Boulder, CO.
12. Samson, P. and J. L. Mallet, Curvature analysis of triangulated surfaces in structural geology, Mathematical Geology **29** (1997), 391–412.
13. Savignat, J. M., A. Rassineux, O. Stab, and P. Villon, High resolution curvature analysis and its applications to surface modeling, preprint, 1999.
14. Savignat, J. M., O. Stab, and A. Rassineux, Reconnaissance des surfaces par analyse de la courbure, Tech. report, École des Mines, Paris, 1999.
15. Shi, P., G. Robinson, and J. Duncan, Myocardial motion and function assesment using 4d images, IEEE Conference on Visualization in Biomedical Computing, 1994.
16. Villon, P., Contribution à l'optimisation, Thèse d'état, Univ. de Techn. de Compiègne, 1991.

J. M. Savignat
École des Mines
35, rue Saint Honoré
77 305 Fontainebleau Cedex, France

savignat@cges.ensmp.fr
stab@cges.ensmp.fr
alain.rassineux@utc.fr
pierre.villon@utc.fr

Discrete Fairing of Curves and Surfaces Based on Linear Curvature Distribution

R. Schneider and L. Kobbelt

Abstract. In the planar case, one possibility to create a high quality curve that interpolates a given set of points is to use a clothoid spline, which is a curvature continuous curve with linear curvature segments. In the first part of the paper we develop an efficient fairing algorithm that calculates the discrete analogon of a closed clothoid spline. In the second part we show how this discrete linear curvature concept can be extended to create a fairing scheme for the construction of a triangle mesh that interpolates the vertices of a given closed polyhedron of arbitrary topology.

§1. Introduction

In many fields of computer-aided geometric design (CAGD) one is interested in constructing curves and surfaces that satisfy aesthetic requirements. A common method to create fair objects is to minimize fairness metrics, but since high quality fairness functionals are usually based on geometric invariants, the minimization algorithms can become computationally very expensive [10].

A popular technique to simplify this approach is to give up the parameter independence by approximating the geometric invariants with higher order derivatives. For some important fairness functionals this results in algorithms that enable the construction of a solution by solving a linear equation system, but such curves and surfaces are in most cases not as fair as those depending on geometric invariants only.

An interesting approach to simplify the construction process without giving up the geometric intrinsics is to use variational calculus to derive differential equations characterizing the solution of a minimization problem. Mehlum [8] used this idea to approximate a minimal energy curve (MEC), which minimizes the functional $\int \kappa''(s)^2 ds$, with piecewise arc segments. In [1] Brunnet and Kiefer exploited the property that a segment of a MEC between two interpolation points satisfies the differential equation $\kappa'' + \frac{1}{2}\kappa^3 = 0$ to speed up the construction process using lookup tables.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 371–380.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

The usage of such differential equations based on geometric invariants can be seen as a reasonable approach to the fairing problem in its own right, and it can be applied to curves as well as surfaces. For planar curves, one of the simplest differential equations is $\kappa'' = 0$. Assuming arc length parameterization, this equation is only satisfied by lines, circles and clothoids. A curvature continuous curve that consists of parts of such elements is called a clothoid spline. Most algorithms for the construction of such curves are based on techniques that construct the corresponding line, circle and clothoid segments of the spline [9]. This is possible for planar curves, but the idea does not extend to surfaces.

In this paper we will first present a fast algorithm to construct an interpolating closed discrete clothoid spline (DCS) purely based on its characteristic differential equation. Our algorithm uses discrete data, because this has been proven to be especially well suited for the efficient construction of nonlinear splines [7]. The efficiency of our construction process is largely based on an algorithm called the indirect approach. This algorithm decouples the curvature information from the actual geometry by exploiting the fact that the curvature distribution itself is a discrete linear spline. Besides the speed of the planar algorithm, it has another very important property: it directly extends to the construction of surfaces!

In Section 2 we give a short review of related work. Section 3 will first give an exact definition of a DCS and then address its efficient computation. Section 4 shows how the planar algorithm extends to surfaces. We construct fair surfaces interpolating the vertices of a given closed polyhedron of arbitrary topology by assuming a piecewise linear mean curvature distribution with respect to a natural parameterization.

§2. Related Work

A very interesting algorithm addressing the problem of constructing fair curves and surfaces with interpolated constraints was presented by Moreton and Sequin. In [10] they minimized fairing functionals, whose fairness measure punishes the variation of the curvature. The quality of their minimal variational splines is extraordinary good, but due to their extremely demanding construction process, the computation time needed is enormous. Their curves and surfaces consisted of polynomial patches.

In contrast to this approach, most fairing and nonlinear splines algorithms are based on discrete data. Malcolm [7] extended the discrete linear spline concept to efficiently calculate a discrete MEC for functional data. In [13] Welch and Witkin presented a nonlinear fairing algorithm for meshes of arbitrary connectivity, based on the strain energy of a thin elastic plate. Recently, Desbrun et al. [3] used the mean curvature flow to derive a discrete fairing algorithm for smoothing of arbitrary connected meshes.

In most fairing algorithms the original fairing functionals are approximated by simpler parameter dependent functionals. In recent years this idea was combined with the concept of subdivision surfaces to create variational

subdivision splines [12,6,4]. Especially [4] should be emphasized here, since our local parameterization needed in Section 4 is largely based on ideas presented there.

Instead of minimizing a functional, Taubin [11] proposed a signal processing approach to create a fast discrete fairing algorithm for arbitrary connectivity meshes. Closely related but based on a different approach is the idea presented by Kobbelt et al. [5], where a uniform discretization of the Laplace operator is used for interactive mesh modeling. In [3] Desbrun et al. showed that a more sophisticated discretization of the Laplace operator can lead to improved results. The direct iteration approach presented later can be seen as a nonlinear generalization of such schemes.

§3. Discrete Clothoid Splines

In this section we show how to interpolate a closed planar polygon using a discretization of a clothoid spline. Although the algorithms extends to open polygons with G^1 boundary conditions in a straightforward manner, we only consider closed curves, because that case directly extends to surfaces.

3.1 Notation and definitions

In the following we denote the vertices of the polygon that has to be interpolated with $P = \{P_1, \ldots, P_n\}$. Let $Q = \{Q_1, \ldots, Q_m\}$ be the vertices of a refined polygon with $P \subset Q$. The discrete curvature at a point Q_i is defined to be the reciprocal value of the circle radius interpolating the points Q_{i-1}, Q_i, Q_{i+1} or 0 if the points lie on a straight line, which leads to the well known formula

$$\kappa_i = 2\frac{det(Q_i - Q_{i-1}, Q_{i+1} - Q_i)}{||Q_i - Q_{i-1}||\,||Q_{i+1} - Q_i||\,||Q_{i+1} - Q_{i-1}||}. \tag{1}$$

Definition 1. *A polygon Q with $P \subset Q$ is called a* discrete clothoid spline *(DCS), if the following conditions are satisfied:*

1) *The interior of each segment is arc length parameterized*
   $||Q_{i-1} - Q_i|| = ||Q_i - Q_{i+1}||$, *whenever* $Q_i \notin P$,
2) *The discrete curvature is piecewise linear:*
   $\Delta^2\kappa_i = \kappa_{i-1} - 2\kappa_i + \kappa_{i+1} = 0$, *whenever* $Q_i \notin P$.

We will construct a DCS using an iteration procedure $Q^k \to Q^{k+1}$ until the above conditions are sufficiently satisfied. The iteration starts with an initial polygon Q^0 that interpolates the P_i (Fig. 1).

3.2 Direct approach

In this approach the location of a vertex Q_i^{k+1} only depends on the local neighborhood $Q_{i-2}^k, \ldots, Q_{i+2}^k$ of its predecessor Q_i^k. To satisfy condition 1 in Definition 1 locally, Q_i^{k+1} has to lie on the perpendicular bisector between

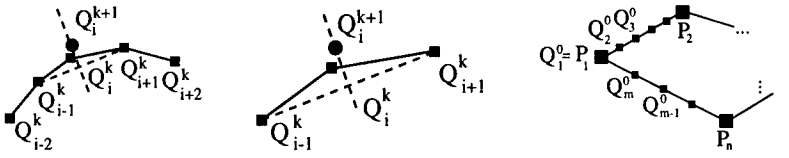


Fig. 1. The left side shows the update steps for Q_i^k in the direct and the indirect iteration. The right side shows the initial polygon Q^0. We simply subsampled the polygon P.

Q_{i-1}^k and Q_{i+1}^k (Fig. 1), reducing the 2 variate problem to a univariate one. Further we require Q_i^{k+1} to satisfy $\Delta^2 \kappa_i^{k+1} = 0$. Unfortunately this equation is nonlinear in the coordinates of the vertices Q^{k+1}, but since the update $Q_i^k \rightarrow Q_i^{k+1}$ in the k-th iteration step will be small, we can linearize the equation by using the coordinates of Q^k instead of Q^{k+1} in the denominator of equation (1). This allows us to update every Q_i^k solving a 2×2 linear system for Q_i^{k+1} during one iteration step.

3.3 Indirect approach

If we decouple the curvature values κ_i^{k+1} from the actual polygonal geometry and perform the direct iteration scheme only on the curvature values by solving $\Delta^2 \kappa_i^{k+1} = 0$, the curvature plot would converge to a piecewise linear function. The idea of the indirect approach is to use this property to create a new iteration scheme. This is done by dividing each iteration step into two sub-steps, in the first sub-step we estimate a continuous piecewise linear curvature distribution, and in the second sub-step we use those as boundary conditions to update the points Q^k. We get the curvature distribution by estimating the curvature of the polygon Q^k at every point P_j, and interpolate this curvature values linearly across the interior of the segments, assigning a curvature estimation $\tilde{\kappa}_i^{k+1}$ to every vertex Q_i^k. In the second step this curvature information is used to update the points Q_i^k, where the position of the new point Q_i^{k+1} is determined by $\tilde{\kappa}_i^{k+1}$ and the neighborhood $Q_{i-1}^k, Q_i^k, Q_{i+1}^k$ of its predecessor Q_i^k (Fig. 1). Again one degree of freedom is reduced by restricting Q_i^{k+1} to lie on the perpendicular bisector between the points Q_{i-1}^k and Q_{i+1}^k, and further we require Q_i^{k+1} to satisfy $\kappa_i^{k+1} = \tilde{\kappa}_i^{k+1}$. Using an analogous linearization technique as in the previous case, we can again update every Q_i^k solving a 2×2 linear system. It is very important to alternate both sub-steps during each iteration. Since the $\tilde{\kappa}_i^{k+1}$ are only estimates of the final DCS, an iteration process that merely iterates the second sub-step might not converge.

Compared to the direct approach, the indirect iteration scheme has some interesting advantages. The interpolation points imply forces everywhere during one iteration step; there is no slow propagation as in the direct approach. The direct approach acts due to its local formulation as a lowpass filter, here because of its global strategy the indirect iteration is much better suited to reduce the low frequency error. One iteration step is cheaper than one iteration step of the direct approach. This fact will become much more important

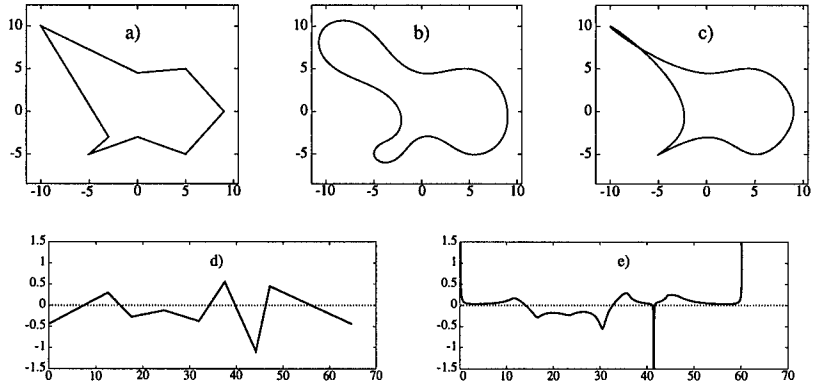


Fig. 2. a) A polygon P with 8 vertices. b) The DCS interpolating the vertices of P. The structure of the polygon is equivalent to a 5 times subdivided P. The calculation time was < 0.1 seconds. c) Periodic C^2 cubic spline interpolation of P. d) Curvature plot of the DCS. e) Curvature plot of the periodic cubic spline.

if we extend the algorithms to the surface case. Finally, since the estimated curvatures are constructed by linear interpolation they are well bounded, a fact that stabilizes the iteration procedure.

3.4 Details and remarks

Both schemes were implemented using a generic multigrid scheme, a technique that has proven to be valuable when hierarchical structures are available [4]. We first constructed a solution on a coarse level, and used a prolongation of this coarse solution as starting point for an iteration on a finer level. Applying this strategy across several levels of the hierarchy, the convergence of both approaches are increased dramatically. On the coarsest hierarchy level the initial polygon was constructed by linearly sampling the polygon P (Fig. 1). The curvature values at the interpolation points $Q_i \in P$ needed in the estimation step were calculated by simply applying formula (1) on Q_i. Without the multigrid approach, such a simple strategy would not be reasonable, because for fine polygons the occurring curvature values could become too large. Our final iteration scheme for the DCS is a combination of the two primary multigrid iterations. Such an approach has the advantage that the estimated curvatures are usually better, since the high frequency error near the interpolation points are smoothed out by the direct step. This multigrid hybrid approach turned out to be a very reliable solver.

All examples throughout the paper were implemented in Java 1.2 for Windows running on a PII with 400MHz. In Figure 2 we see that our algorithm still produces a fair solution in cases where classical periodic cubic spline interpolation using chord length parameterization fails completely. In this example we can also see why it is not only comfortable to be able to start with such a simple initial polygon Q^0, but also can be very important in some

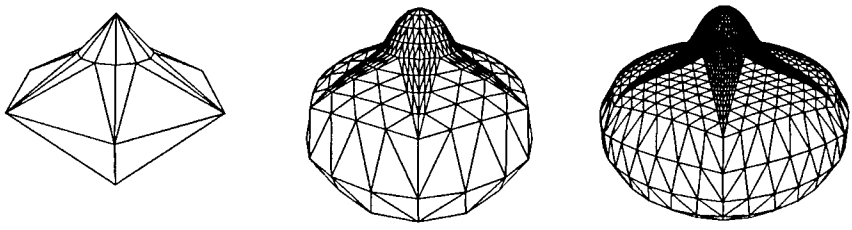


Fig. 3. Wireframe of a mesh P and the solutions of a 2 times resp. 3 times subdivided mesh. Iteration times: middle $\approx 0.2s$, right $\approx 0.4s$.

cases. Our algorithm searches the solution next to the starting polygon, so taking Q^0 to be a a linearly sampled P will prefer solutions without loops. A technique that would use the periodic cubic spline to initialize Q^0 in this example would either fail because of the enourmous curvatures that turn up, or produce a DCS with loops.

It is well known that clothoids can be parameterized using Fresnel Integrals [9]. If such a continuous solution is preferred, the discrete solution could be used to derive a closed form representation.

§4. Closed Surfaces of Arbitrary Topology

In this section we want to show how to extend the concept of planar clothoid splines to closed surfaces of arbitrary topology. Given a closed polyhedron P of arbitrary topology, we construct a discrete fair surface that interpolates the vertices of P. To be able to extend the planar algorithms, we first have to determine what curvature measure for surfaces has to be used. According to Bonnet's uniqueness problem, the mean curvature seems to be most appropriate.

4.1 Notation and definitions

In the following, we denote the vertices of the polyhedron that has to be interpolated by $P = \{P_1, \ldots, P_n\}$. Let $Q = \{Q_1, \ldots, Q_m\}$ be the vertices of a refined polyhedron with $P \subset Q$, where we require the topological mesh structure of Q to be equivalent to a uniformly subdivided P (Fig. 3). Because of this structure, we can partition the vertices of Q into three classes. Vertices $Q_i \in P$ are called interpolation vertices, Q_i that can be assigned to an edge of P are called edge vertices, and the remaining points are called inner vertices. Edge and inner vertices Q_i always have valence 6; for those points let $Q_{i,l}, l = 1..6$ be their adjacent vertices. For edge vertices we make the convention that the adjacent vertices are arranged such that $Q_{i,1}$ and $Q_{i,4}$ are edge or interpolation vertices. Let H_i resp. $H_{i,l}$ be the discrete mean curvature at Q_i resp. $Q_{i,l}$ and let n_i be the discrete unit normal vector at Q_i. Finally, let us define the following operators:

$$g_p(Q_i) = \begin{cases} 1/6 \sum_{l=1}^{6} Q_{i,l}, & \text{if } Q_i \text{ is an inner vertex,} \\ 1/2\,(Q_{i,1} + Q_{i,4}), & \text{if } Q_i \text{ is an edge vertex,} \end{cases}$$

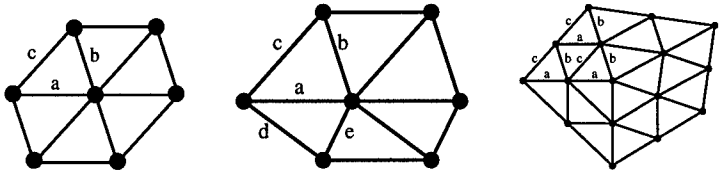


Fig. 4. Left) Parameter domain for an inner vertex. Middle) Parameter domain for an edge vertex. Right) The parameter domain for an interpolation vertex of valence 5 is a subset of this domain.

$$g_h(H_i) = \begin{cases} 1/6 \sum_{l=1}^{6} H_{i,l}, & \text{if } Q_i \text{ is an inner vertex,} \\ 1/2\,(H_{i,1} + H_{i,4}), & \text{if } Q_i \text{ is an edge vertex.} \end{cases}$$

We are now in the position to extend Definition 1 to the surface case.

Definition 2. *The polyhedron Q will be called a solution to our discrete fairing problem, if the following conditions are satisfied:*

1) *The vertices Q_i are regularly distributed. For all edge and inner vertices, there should be a $t_i \in \mathbb{R}$ such that $Q_i = g_p(Q_i) + t_i \vec{n}_i$,*
2) *The curvature is linearly distributed over each face: $g_h(H_i) = H_i$.*

Condition 2 is straightforward, it states that the mean curvature is changing linearly with respect to the barycentric parameterization of the inner and edge vertices. Condition 1 is more difficult to understand. It generalizes the DCS property that vertices in the interior of a segment were on the perpendicular bisector between its neighbors.

As in the planar case, we construct the surfaces using an iteration $Q^k \rightarrow Q^{k+1}$, starting with an initial polyhedron Q^0.

4.2 Discretization of the geometric invariants

Our discretization technique is based on the well-known idea of constructing a local quadratic least square approximation of the discrete data and estimating the needed geometric invariants from the first and second fundamental forms. For the mean curvature H, this means [2]

$$H = \frac{1}{2} \frac{eG - 2fF + gE}{EG - F^2}, \tag{2}$$

where E, F, G are the coefficients of the first fundamental form, and e, f, g are those of the second fundamental form of the least square approximation.

Instead of recalculating a local parameterization during each step in the iteration, we exploit the fact that our meshes are well structured to determine a local parameterization in advance, thus raising the speed of our algorithms considerably. The decision, which local parameterization should be assigned to a vertex Q_i was influenced by three mayor constraints: simplicity, regularity and uniqueness of the quadratic approximation. These constraints lead us to the following local parameterization classes (Fig. 4). For inner vertices, the

parameter domain is a regular hexagon, for edge vertices it is composed of two regular hexagon halves. Calculating the matrices needed for a least square approximation in both cases, it is easy to see that this parameterizations always lead to a unique least square approximation. Only at the interpolation vertices it is not always sufficient to use the 1-neighborhood. This is obvious if the valence v of the vertex is 3 or 4, but even if the valence is higher, the least square approximation can fail in the 1-neighborhood. Using parts of the 2-neighborhood consisting of regular sectors solves that problem. For example, the least square approximation is already unique by using the 1-neighborhood and one complete sector of the 2-disc (marked with a,b,c in Fig. 4).

Since we are only interested in intrinsic values, we can use the fact that an affine mapping of the parameter domain only changes the parameterization of our quadratic approximation, but not geometric invariants. This fact allowed us to chose one fixed equilateral hexagon to serve as parameter domain for all inner vertices, and guaranteed a simple update step for such points.

At edge and interpolation vertices, we calculated the parameter domains by applying the blending technique proposed in [4]. This algorithm constructs a local parameterization that adapts to the underlying geometry defined by P, thus approximating a local isometric parameterization. For a detailed description of that algorithm we have to refer to that work.

4.3 Direct approach

When updating the vertex Q_i^k in an iteration step, the new position is determined by the two equations $Q_i^{k+1} = g_p(Q_i^k) + t_i \vec{n}_i^k$ and $H_i^{k+1} = g_h(H_i^k)$. With analogous arguments as in the curve setting, we linearized the mean curvature expression (2) for H_i^{k+1} by using the vertices of Q^k to calculate the coefficients of the first fundamental form E, F, G and by replacing the normal n_i^{k+1} by n_i^k when calculating the coefficients of the second fundamental form. Finally, the vertex Q_i^{k+1} is determined by solving a linear equation for t_i.

4.4 Indirect approach

The planar indirect iteration scheme directly extends to the surface setting. In the first sub-step we estimate the mean curvature of the polyhedron Q^k at the interpolation vertices and use simple linear interpolation to assign a mean curvature value $\tilde{H}_i^{k+1}$ to every edge and inner vertex Q_i^k. This means the estimated values satisfy $\tilde{H}_i^{k+1} = g_h(\tilde{H}_i^{k+1})$. In the second sub-step we then use these estimated values to determine Q_i^{k+1} using the formula $H_i^{k+1} = \tilde{H}_i^{k+1}$ with the same linearization method as in the direct approach.

4.5 Details and remarks

Since the vertices are updated along the surface normals, the direct iteration can be interpreted as some kind of curvature flow, where the speed is determined by a local equation system. As was pointed out by Desbrun et al. [3], flows depending only on local surface properties do not have to be numerically stable for large steps during the iteration process. To allow larger

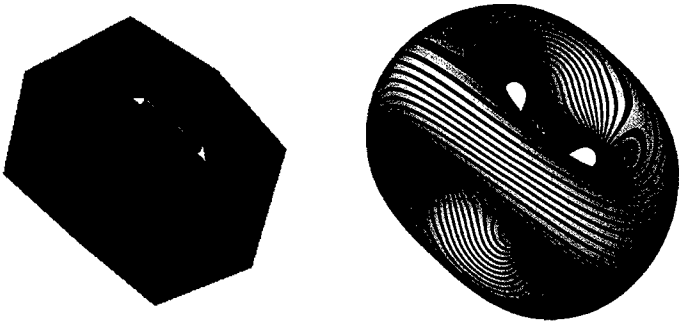


Fig. 5. Left) Tetra Thing mesh. Right) Solution of the multigrid indirect iteration after the Tetra Thing has been subdivided 5 times. The reflection lines indicate that the surface is quasi G^2 continuous. Iteration time: ≈ 4 seconds.

update steps, Desbrun et al. used the backward Euler method to develop an algorithm called implicit integration for fairing of arbitrary meshes based on Laplacian and on mean curvature flow. The idea behind this approach is to use global surface information instead of considering the local neighborhood only. The indirect approach follows that idea and exploits global surface properties efficiently.

In the surface case the indirect approach shows its true power. The estimation sub-step is mostly simple linear interpolation and using the estimated curvature, we only need the 1-neighborhood of a vertex during its update process. The update process for interior vertices is especially simple because of the equilateral hexagon as parameter domain. For edge and interpolation vertices, we calculated the least square matrices before the first iteration step and cached the result. Our surface examples were created using a mutigrid iteration scheme based on the indirect approach.

As mentioned earlier, the construction of our local parameterization is based on the technique presented in [4], where divided difference operators to create discrete thin plate splines had to be derived. A comparison of our results showed that the ideas presented in this paper allow the construction of considerably improved meshes without increasing the computation time. The improved quality is due to the fact that our discrete surfaces are based on geometric invariants instead of second order partial derivatives, and thus the error made by guessing an isometric local parameterization in advance has less influence. Due to the fact that we estimated the local parameterization once, we do not get an exact G^2 continuity at the edges and interpolating points, but as can be seen in Figure 5, our surfaces are good approximations to G^2 surfaces. To achieve true G^2 continuity, one would have to increase the effort for the handling of edge and interpolating vertices. Future work will investigate if such efforts are worthwhile.

§5. Conclusion

The quality of the resulting curves and surfaces is superior to approaches based on quadratic functionals as the thin plate energy due to the more sophisticated approximation of geometric invariants. Although our objects are nonlinear splines, they can be calculated fast enough to be applicable in interactive design. The presented ideas are optimally suited for the construction of curves and surfaces with piecewise linear curvature distribution, but could also be extended to higher order discrete fairing approaches.

References

1. Brunnett G., and J. Kiefer, Interpolation with minimal-energy splines, Computer-Aided Design **26**(2) (1994), 137–144.
2. do Carmo, M. P., *Differential Geometry of Curves and Surfaces* Prentice-Hall, Inc Englewood Cliffs, New Jersey, 1993.
3. Desbrun, M., M. Meyer, P. Schröder, and A. H. Barr, Implicit fairing of irregular meshes using diffusion and curvature flow, *SIGGRAPH 99 Conference Proceedings*, 317-324.
4. Kobbelt, L., Discrete fairing, Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces '97, 101–131.
5. Kobbelt, L., S. Campagna, J. Vorsatz, and H. P. Seidel, Interactive multi-resolution modeling on arbitrary meshes, *SIGGRAPH 97 Conference Proceedings*, 105–114.
6. Kobbelt, L., A variational approach to subdivision, Comput. Aided Geom. Design **13** (1996), 743–761.
7. Malcolm M. A., On the computation of nonlinear spline functions, SIAM J. Numer. Anal. **15** (1977), 254–282.
8. Mehlum, E., Nonlinear splines, *Computer Aided Geometric Design*, Academic Press, London, 173–207, 1974.
9. Meek, D. S., and D. J. Walton, A guided clothoid spline, Comput. Aided Geom. Design **8** (1991), 163–174.
10. Moreton H. P., and C. H. Séquin, Functional optimization for fair surface design, *SIGGRAPH 92 Conference Proceedings*, 167–176.
11. Taubin G., A signal processing approach to fair surface design, *SIGGRAPH 95 Conference Proceedings*, 351–358.
12. Weimer, H., and J. Warren, Subdivision Schemes for Thin Plate Splines, Computer Graphics Forum **17** (1998), 303–314.
13. Welch, W., and A. Witkin, Free-Form shape design using triangulated surfaces, *SIGGRAPH 94 Conference Proceedings*, 247–256.

Leif Kobbelt and Robert Schneider
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany
{kobbelt, schneider}@mpi-sb.mpg.de

Triangulating Trimmed NURBS Surfaces

Chang Shu and Pierre Boulanger

Abstract. This paper describes techniques for the piecewise linear approximation of trimmed NURBS surfaces. The problem, called surface triangulation, arises from many applications in CAD and graphics. The new method generates triangular meshes that are adaptive to the local surface curvature. We use efficient data structures for the handling of trimming curves. We also generate Delaunay triangulation on the surface to improve the quality of the meshes.

§1. Introduction

Tensor-product NURBS are widely used in today's CAD systems for describing and exchanging surface geometry. For many applications, however, piecewise linear approximations of smooth surfaces are required. Examples of these applications include finite element analysis, stereo-lithography, and visualization of geometric models. In these applications, we need to generate a triangular mesh that approximates the original surface within a given tolerance. We refer to this problem as surface triangulation, stated in the following definition.

Definition. *Given a NURBS surface $N(u,v)$, its trimming boundary, and a real number ε, the* surface triangulation problem *is to find a set of linearly parameterized triangles $\{T_i\}$ such that*

1) *Any triangle T_i satisfies* $\sup \|T_i(u,v) - N(u,v)\| < \varepsilon$.
2) *For any triangle edge not on the boundary, there is exactly one neighboring triangle sharing this edge.*

The first condition is usually called chord height tolerance, which restricts triangles to be close to the surface. The second condition requires the triangular mesh to be topologically correct.

A good surface triangulation algorithm is expected to be efficient because real world models tend to contain large numbers of surface patches. Furthermore, certain optimization factors are desirable. Two of the most important ones are triangle shape and the number of triangles in the mesh. For example, in finite element analysis, triangles with bad aspect ratio (one angle is

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 381–388.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

significantly smaller or larger than the others) reduce the solution precision. In all applications, a mesh with a small number of triangles saves computing and transmission time as well as storage space.

Several authors [4,6,8,14] have approached the surface triangulation problem by computing a bound on the length of triangle edges in parametric space so that if all triangles have their edge lengths smaller than the bound, the resulting triangulation satisfies the chord height tolerance. Since the edge length bound applies to the whole surface, the density of the triangulation distributes uniformly across the surface and may lead to unnecessarily large mesh size. Along another line of thought, Klein and Straßer [5] considered the problem of placing points based on the surface curvature. Recently, Piegl and Tiller [11] used adaptive subdivision of the surface. Obviously, for a given chord height tolerance, adaptive algorithms generate fewer triangles than the uniform subdivision algorithms. But adaptive algorithms tend to be slower.

In this paper, we give a method that has the following features:

1) adaptive to the surface curvature,
2) efficient insertion of the trimming curves,
3) triangle shape improvement.

Our general strategy is that we first approximate the surface with hierarchical quadrilaterals without considering the trimming curves, then we insert the trimming curves and triangulate the quadrilaterals. The result is a triangulation that satisfies the chord height tolerance. We improve the efficiency of trimming curve insertion by organizing the quadrilateral hierarchies in a quadtree structure. Also, we improve the quality of the triangles by converting the initial triangulation to a Delaunay triangulation.

§2. Curve and Surface Subdivision

We begin by discretizing the surface and its trim curves independently. We assume that the surfaces are trimmed in the parametric domain and the trimming curves are represented as NURBS with consistent orientation. Our first objective is to approximate the curves with connected line segments such that they do not deviate from the surface more than the tolerance ε. From well-known results in B-Spline theory [7,10], a NURBS curve can be split into two pieces without changing its shape by inserting new knots. The consequence of this splitting is that we introduce new control points that are closer to the curve than the control points of the original curve. If we keep dividing in this way, the control polygon converges to the surface. When a sub-curve's control polygon becomes "flat" enough, we can stop the dividing process. According to the convex hull property of NURBS, the maximum distance from any point on the sub-curve to the line segment joining the two end control points is bounded by the maximum distance between control points to the segment. Therefore, we can use this bound to control the flatness of the sub-curves.

The surface can be approximated in the same way by quadrilaterals. At this time, we ignore the trimming boundary. Here, we insert knots in both u

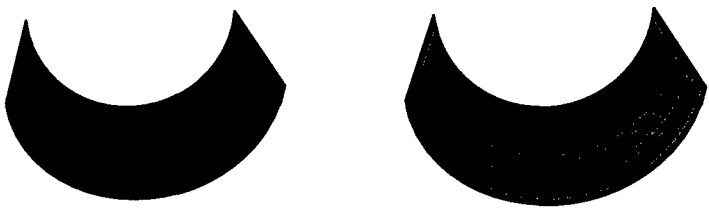


Fig. 1. Full surface subdivision.

and v directions. The flatness test is a little more complex. We examine every row and column of the control polygon and test their flatness. We also have to consider the twist factor of a surface patch. Peterson [9] gives a subdivision method, which we generally follow. Fig. 1 (left) shows a surface approximated by quadrilaterals.

We use a quadtree data structure to keep track of the surface subdivision process. A quadrilateral is divided if it does not satisfy the flatness test. Its children are subject to the same test until at a certain level they are flat enough. Therefore, more subdivisions are needed at places where surface curvature is high.

§3. Trim Curve Insertion

We assume the trimming curves are given in the parametric space of the surfaces which they delineate. They are first discretised into line segments using the same tolerance for the surface. Then the trimming curve segments are inserted into the quadtree cells by walking through the quadrilaterals using adjacency information. The right-hand figure in Fig. 1 shows an example of the insertion.

The insertion can be done completely in the parametric domain in which the quadrilaterals correspond to rectangles in two dimensions. Starting with a vertex of the trimming segments, we first find the rectangle in which this vertex is contained. This can be done efficiently by traversing down the quadtree. By following the trimming segments, we can find the segment that crosses one of the edges of the rectangle. We insert a new vertex on the intersection point and then start the insertion in the new rectangle.

For efficient insertion of the trimming segments, we make use of a data structure that can quickly find the neighboring rectangle from the edge of a rectangle. We store in each rectangle (quadrilateral) vertex the pointers of the rectangles that use the vertex. Given an edge $e = (p, q)$, we collect all the rectangles that contain both p and q, $Q_e = Q_p \cap Q_q$, where Q_p and Q_q are the sets of rectangles associated with vertices p and q respectively. This is a local operation. The number of elements in Q_e should either be 1 or 2.

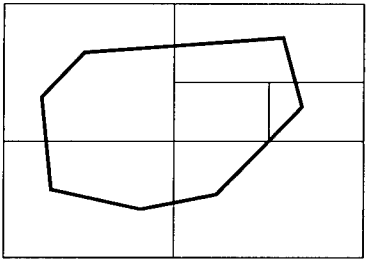


Fig. 2. Trimming curve insertion.

In the case of two elements, one of them is the neighboring rectangle we look for. When there is only one element in Q_e, our rectangle does not have a compatible neighbor. However, if we go up the quadtree, at some level there must be a rectangle that is the compatible neighbor. Then coming down the tree, we find the leaf rectangle that is partially neighboring our initial quad. This is the rectangle that the trimming segment enters.

The time complexity of the insertion process is linear in the number of quadrilaterals.

Fig. 2 shows the insertion of trimming curves into the rectangles in parametric domain. Fig. 3 shows the two cases that a trimming segment enters a new rectangle: 1) entering from an edge; 2) entering from a vertex.

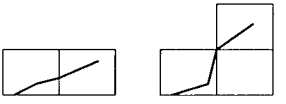


Fig. 3. Two entering cases.

§4. Initial Triangulation

After the insertion of the trimming segments, we have two kinds of rectangles: those that are cut by the trimming segments and those that do not intersect with any part of the trimming segments. For each rectangle being cut, we sort the vertices of the trimming segments inside the rectangle in counter-clockwise order to form boundaries of polygons. In general, there can be multiple polygons and each polygon can have multiple boundary loops. Those cells that lie inside the boundary are triangulated in parametric space. There is no shortage of triangulation algorithms for 2-dimensional polygonal domains. Here we adopt the algorithm from [13]. Note that most rectangles lie completely in the interior of the trimming boundary; their triangulation can simply be done by triangulating a rectangular domain [1]. If a rectangle is cut-free and lies outside the trimming boundary, we can simply ignore it. This case can be decided easily by testing if one of the vertices of the rectangle is in the interior of the polygon formed by the trimming segments. For robustness reasons, we choose the centroid of the rectangle for doing the test.



Fig. 4. Initial triangulation in parametric space and in 3-space.

An initial triangulation in 3-space is obtained by evaluating the parametric triangulation. Fig. 4 gives an example of a surface triangulated in the parametric domain (left) and the corresponding triangulation in 3-space (right).

§5. Triangle Shape Improvement

As we mentioned in Section 1, there are good reasons to make triangles that are well-shaped. In practice, it is undesirable to have triangles that are flat or pointed. These are the triangles that have one small angle or one large angle. It is well known that Delaunay triangulation for a set of points in two dimensions is optimal in the sense that it maximizes the minimum angle [2]. Delaunay triangulation that respect a set of boundary edges can be constructed. This kind of triangulation is called constrained Delaunay triangulation (CDT) [12]. Chew [3] extended the definition of CDT to the curved surfaces by replacing the empty circumcircle condition with the empty minimum circumsphere condition. Following Chew's approach, we improve the shape of the triangles by edge flipping and inserting new nodes at the circumcenters of the ill-shaped triangles.

Given a pair of triangles, if they form a convex quadrilateral, there are two choices of the diagonals, one is better than the other in terms of the shapes of the triangles. By examining each pair of adjacent triangles and flipping their diagonals if necessary, we can improve the triangulation locally (see top figures of Fig. 5). Chew [3] shows that the flipping process halts and it leads to constrained Delaunay triangulation on a surface.

A CDT is the best possible triangulation without introducing new nodes. To further improve the triangulation, we have to insert new points. Each time we insert a new point, we do edge flipping again to maintain Delaunay triangulation. New points are inserted at the circumcenters of the triangles that violate the shape criteria. The reason of this is we could improve the shape of several triangles by introducing one point. Fig. 5 illustrates the two basic operations used repeatedly for improving the shape of the triangles.

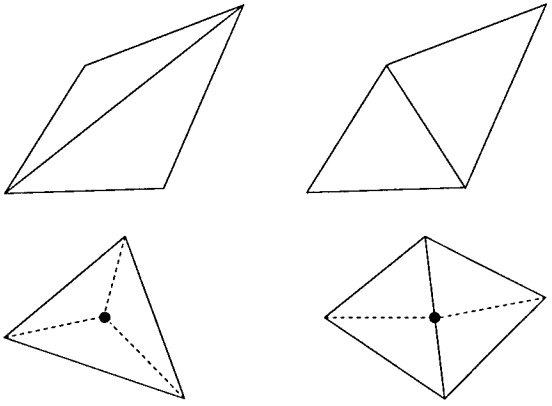


Fig. 5. Edge flipping and node insertion.

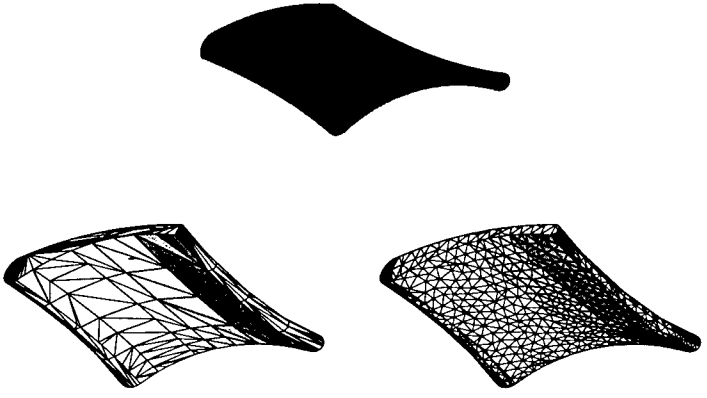


Fig. 6. Example 1.

As we flip edges, we want to preserve the error bound for the new triangulation. Given a pair of triangles that are flippable, we check the minimum distance between the current diagonal and the new diagonal. The distance should be smaller than the specified approximation tolerance ε. This does not guarantee that the resulting triangulation still satisfies the approximation tolerance. But since we are not moving any nodes on the surface and we insert additional nodes into the triangulation, there are good reasons to assume that most triangles will satisfy the tolerance. Finally, as a last step, we loop through all triangles and check their chord heights. For those few triangles that violate chord height tolerance, we subdivide them by adding points on their edges. The checking is generally expensive, but we only do this once

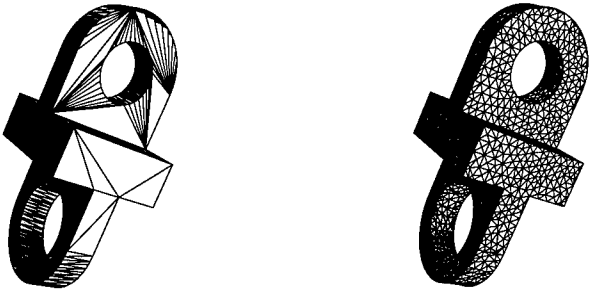


Fig. 7. Example 2.

for each triangle, and the process can be speeded up by making use of the quadtree data structure.

Figs. 6 and 7 give two results of the algorithm before and after shape improvements.

§6. Concluding Remarks

The main results of this work are:

1) a surface triangulation algorithm that guarantees correct mesh topology,
2) an efficient trimming curve insertion scheme,
3) triangle shape improvement by Delaunay triangulation.

We have only discussed the problem of triangulating a single surface. However, in real world problems, a model usually consists of many NURBS surface patches. The triangulation between two neighboring surfaces have to be compatible. There should be a post-processing step that stitches the triangulation of different surfaces. This can be done with a kd-tree data structure, which facilities locating nearest nodes in 3-space quickly. Therefore, we can propagate a node on the boundary of one surface to the boundary of its neighboring surface.

References

1. Baehmann, P. L., S. L. Wittchen, M. S. Shephard, K. R. Grice and M. A. Yerry, Robust, geometrically based, automatic two-dimensional mesh generation, International Journal for Numerical Methods in Engineering **24** (1987), 1043–1078.
2. Bern, M. and D. Eppstein, Mesh generation and optimal triangulation, in *Computing in Euclidean Geometry, 2nd ed.*, D.-Z. Du and F. K. Hwang, (eds.), World Scientific, 1995, 47–123.
3. Chew, P., Guaranteed-quality mesh generation for curved surfaces, Proc. of the 9th Annual Symposium on Computational Geometry, 1993, 115–127.

4. Filip, D. J., R. Magedson and R. Makot, Surface algorithms using bounds on derivatives, Comput. Aided Geom. Design **3** (1986), 295–311.
5. Klein, K. and W. Straßer, Large mesh generation from boundary models with parametric face representation, Proc. of the 3rd ACM Symposium on Solid Modeling and Applications, 1995, 431–440.
6. Kumar, S. and D. Manocha, Efficient rendering of trimmed NURBS surfaces, Computer-Aided Design **27** (1995), 509–521.
7. Lane, J. M. and R. F. Riesenfeld, A theoretical development for the computer generation and display of piecewise polynomial surfaces, IEEE Trans. Pattern Analysis and Machine Intelligence **2** (1980), 35–46.
8. Lane, J. M. and R. F. Riesenfeld, Bounds on a polynomial, BIT **21** (1981), 112-117.
9. Peterson, J. W., Tessellation of NURBS surfaces, in *Graphics Gems IV*, P. S. Heckbert (ed.), Academic Press, New York, 1994, 286–320.
10. Piegl, L. and W. Tiller, *The NURBS Book*, Springer-Verlag, 1997.
11. Piegl, L. and W.Tiller, Geometry-based triangulation of trimmed NURBS surfaces, Computer-Aided Design **30** (1998), 11–18.
12. Ruppert, J., A Delaunay refinement algorithm for quality 2-dimensional mesh generation, Journal of Algorithms **18** (1995), 548–585.
13. Seidel, R., A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, Computational Geometry: Theory and Applications **1** (1991), 51–64.
14. Sheng, X. and B. E. Hirsch, Triangulation of trimmed surfaces in parametric space, Computer-Aided Design **24** (1992), 437–444.

Chang Shu and Pierre Boulanger
Institute for Information Technology
National Research Council of Canada
Montreal Road, Ottawa, Ontario
Canada K1A 0R6
chang.shu@iit.nrc.ca, pierre.boulanger@iit.nrc.ca

Surface Interpolation of Non-four-sided and Concave Area by NURBS Boundary Gregory Patches

Junji Sone, Kouichi Konno, and Hiroaki Chiyokura

Abstract. Subdivision methods are widely used for surface interpolation of a non-four-sided area. Using this method, smooth surface shape control of multiple surfaces is difficult. Therefore, we used the trim surface for interpolation of a concave area, but shape control of the trim surface was also difficult. In this research, the surface interpolation method of non-four-sided (over 4 sided) area uses a single NBG (NURBS Boundary Gregory) patch. Moreover we apply this method for concave areas and study the ability of interpolation. Interpolation of one-sided and two-sided concave shapes is also considered. As a result, one-sided concave area can be interpolated smoothly for complex cases. Two-sided cases need further study.

§1. Introduction

Catmull-Clark [1] and Doo-Sabin [5] subdivisions are widely used to interpolate a non-four-sided area. This method increases the number of surfaces. Designers must modify the non-four-sided area, observing the contour curves, silhouette pattern and highlight lines [13] when modeling the outer shape of products. Surface shape control is more difficult for multiple surfaces, which are generated by subdivision, because it corrects several surfaces at the same time. Moreover, if we apply Catmull-Clark subdivision to concave areas, part of the inner surface twists and protrudes from the boundary curves. Then, the trimmed surface is commonly used for concave area interpolation. Boundary curve of trimmed surface must coincide with inner surface shape. If we modify the trimmed boundary curves, surface shape must follow the change in boundary curves precisely. This modification is very difficult. The Vertex Blending method [6,2] is proposed to interpolate the non-four-sided area by a single patch. It is also difficult to apply this method for concave area interpolation.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 389–398.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

In this paper, we propose the surface interpolation method using a single NURBS boundary Gregory(NBG) [14,8] patch for non-four-sided areas and concave areas. Applying this method to concave area interpolation, selection of the four edges is important. Our basic idea is that smooth streamlines make smooth surfaces. Smoothness of streamlines would be influenced by the construction of four edges. Therefore, we propose the evaluation parameters to evaluate the smoothness of streamlines. Then, we develop the edge construction method using evaluation parameters and apply this method to one and two-sided of the concave area. As a result, single NBG patch can interpolate non four-sided-area, including concave shapes, allowing the surface shape to be modified flexibly.

§2. Surface Construction of NBG Patch for Non-four-sided Areas

2.1 Abstract of NURBS boundary Gregory patch

Boundary curves of the NBG patch are expressed as a NURBS [15] curve. This patch is an extended general boundary Gregory patch [9], constructed from three sub-patches and calculated by following equation:

$$\mathbf{S}(u,v) = \mathbf{S}^u(u,v) + \mathbf{S}^v(u,v) - \mathbf{S}^c(u,v). \tag{1}$$

$\mathbf{S}^u$ is defined by the boundary curves $\mathbf{S}(0,v)$ and $\mathbf{S}(1,v)$ and their cross boundary derivatives $\mathbf{S}_u(0,v)$ and $\mathbf{S}_u(1,v)$. The boundary curves and the cross boundary derivatives are expressed by a NURBS. Similarly, $\mathbf{S}^v$ is represented by boundary curves $\mathbf{S}(u,0)$ and $\mathbf{S}(u,1)$ and the cross boundary derivatives $\mathbf{S}_v(u,0)$ and $\mathbf{S}_v(u,1)$. $\mathbf{S}^c$ is called a common surface $\mathbf{S}^u$ and $\mathbf{S}^v$, which is a cubic rational boundary Gregory patch [3]. Detailed surface construction is described in the reference paper [14,8].

2.2 Abstract of surface interpolation of non-four-sided areas

In this subsection, a surface interpolation method using NBG patch for non-four-sided (over 4 sided) areas is described. We explain the continuity correction method in the next section.

2.2.1 Sub-patch generation of $\mathbf{S}^u$

Fig. 1 shows an interpolation method of pentagonal area. Here the u-direction order of $\mathbf{S}^u$ and the v-direction order of $\mathbf{S}^v$ are cubic. The subdivision method of $\mathbf{S}^u$ is described in this figure. First, the new point $\mathbf{P}_{12}$ is generated by dividing the edge at the same parameter point of C^0 vertex $\mathbf{P}_{11}$. Next, a straight line is generated from $\mathbf{P}_{11}$ to $\mathbf{P}_{12}$. The vector $\mathbf{V}^{c1}$ is calculated by multiplying 1/3 to the vector from $\mathbf{P}_{11}$ to $\mathbf{P}_{12}$. The plane $\mathbf{PL}^1$ is generated from the tangent vector of boundary curves $\mathbf{V}^{u11}$ and $\mathbf{V}^{u21}$. $\mathbf{V}^{u18'}$ is derived by projecting $\mathbf{V}^{c1}$ to plane $\mathbf{PL}^1$. If continuity is more than G^1 at point $\mathbf{P}_{12}$, $\mathbf{V}^{u17'}$ is calculated by a weighted average of the tangent vector $\mathbf{V}^{u14}$ and $\mathbf{V}^{u24}$ using the $\mathbf{P}_{12}$ parameter value. Occasionally, the length of $\mathbf{V}^{u17'}$ and $\mathbf{V}^{u18'}$

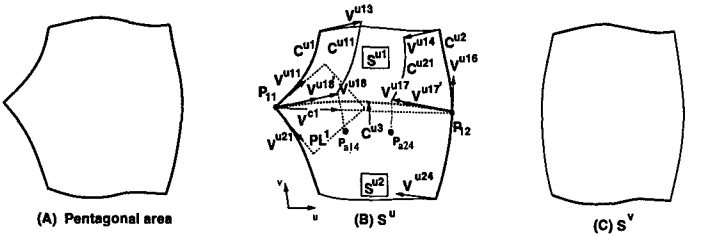


Fig. 1. Surface interpolation method for pentagonal area using a NBG patch.

are much different. In this case, the control polygon decided by $\mathbf{P}_{11}, \mathbf{P}_{12}$,, $\mathbf{V}^{u18'}, \mathbf{V}^{u17'}$ is compensated as equal lengths and new points $\mathbf{V}^{u18}, \mathbf{V}^{u17}$ are generated. The cubic Bézier curve $\mathbf{C}^{u3}$ is generated from the derived points and $\mathbf{S}^u$ is subdivided.

The sub-patch generation procedure is as follows. Firstly, the boundary curves $\mathbf{C}^{u1}$ and $\mathbf{C}^{u2}$ are converted to NURBS curves. The order of $\mathbf{C}^{u1}$ and $\mathbf{C}^{u2}$ are adjusted to the same value. Secondly, a quadratic Bézier form derivative function is generated from the vectors $\mathbf{V}^{u13}$, $\mathbf{V}^{u18}$ and the average of these vectors. This function is converted to NURBS, the degree is elevated to the same order, and knots are inserted as the same knot vectors of boundary curve $\mathbf{C}^{u1}$. Thirdly, $\mathbf{C}^{u11}$ can be derived by adding this derivative function to the boundary curve $\mathbf{C}^{u1}$. Similarly, $\mathbf{C}^{u21}$ can be computed from the vectors $\mathbf{V}^{u14}$,$\mathbf{V}^{u17}$ and the average of these vectors. Finally, the NURBS surface whose u-direction order is 4, is generated from four NURBS curves $\mathbf{C}^{u1}$,$\mathbf{C}^{u11}$, $\mathbf{C}^{u21}$ and $\mathbf{C}^{u2}$.

2.2.2 Continuity correction between sub-patches

Continuity of sub-patch $\mathbf{S}^{u1}$ and $\mathbf{S}^{u2}$ must be contained in G^1 for constructing G^1 continuity NBG patch. The quadratic derivative function is calculated from the tangent vectors $\mathbf{V}^{u11}, \mathbf{V}^{u16}$ of the boundary curve $\mathbf{S}^{u1}$ and the average of the tangent vectors $\mathbf{C}^{u11}$ and $\mathbf{C}^{u21}$. The inner control points of the sub-patch $\mathbf{S}^{u2}$ are modified using the G^1 connection method [4]. Through this procedure, the continuity of the sub-patches $\mathbf{S}^{u1}$ and $\mathbf{S}^{u2}$ can be made G^1. Finally a single NURBS surface $\mathbf{S}^u$ is generated by concatenating these sub-patches.

2.3 G^1 connection with surrounding surface

The non-four-sided surface is generated at the intersection area of three or more fillets. This non-four-sided surface must be joined to the surrounding surface with G^1 continuity. Sarraga [11] shows the G^1 continuity and twist compatibility condition around the vertex where N surfaces are joined. This research is limited to the pair of boundary curves that must be joined with G^1 continuity. However, in our case, the pair of boundary curves is not joined G^1. Therefore, we propose the blending method of twist vectors to satisfy

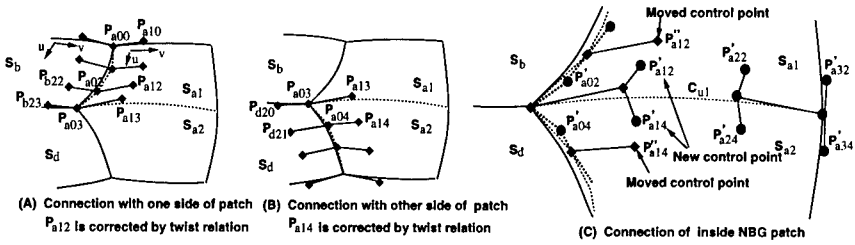


Fig. 2. G^1 connection with surround surface.

the twist compatibility conditions. The procedure to make the G^1 connection with surrounding surface is described below.

2.3.1 Cubic boundary curve

Fig. 2 shows the continuity correction method around a C^0 vertex. We explain that boundary curves are cubic NURBS and their weights are 1.0 which is the same as the cubic Bézier curve. The surface interpolation method for non-four-sided area $\mathbf{S}_a$ is described as follows: Firstly, the continuity between $\mathbf{S}_a$ and $\mathbf{S}_b$ is corrected to G^1 using quadratic Bézier cross boundary derivatives. The basic G^1 connection equation is expressed as

$$\frac{\partial \mathbf{S}_{a1}(u,0)}{\partial v} = k(u)\frac{\partial \mathbf{S}_b(u,1)}{\partial v} + h(u)\frac{\partial \mathbf{S}_{a1}(u,0)}{\partial u}. \tag{2}$$

Here, $k(u) = k_0(1-u) + k_1 u, h(u) = h_0(1-u) + h_1 u$.

The twist equation is generated by differentiating equation (2) with respect to u as follows:

$$\begin{aligned}\frac{\partial^2 \mathbf{S}_{a1}(u,0)}{\partial v \partial u} = {} & k_u(u)\frac{\partial \mathbf{S}_b(u,1)}{\partial v} + k(u)\frac{\partial^2 \mathbf{S}_b(u,1)}{\partial v \partial u} + \\ & h_u(u)\frac{\partial \mathbf{S}_{a1}(u,0)}{\partial u} + h(u)\frac{\partial^2 \mathbf{S}_{a1}(u,0)}{\partial^2 u}.\end{aligned} \tag{3}$$

The control point $\mathbf{P}_{a12}$ can be regenerated using twist compatibility equation (3). $\mathbf{P}_{a12}$ is calculated by applying the quadratic derivative function to equation (3) as follows:

$$\begin{aligned}\mathbf{V}_{a2} = {} & \frac{1}{3}\Big[3\mathbf{V}_{a3} - \frac{n_{vb}(k_1-k_0)\mathbf{V}_{b2}}{3} + n_{vb}k_1(\mathbf{V}_{b2}-\mathbf{V}_{b1}) \\ & + (h_1-h_0)\mathbf{V}_{c2} + 2h_1(\mathbf{V}_{c2}-\mathbf{V}_{c1})\Big], \\ \mathbf{P}_{a12} = {} & \mathbf{V}_{a2} + \mathbf{P}_{a03}.\end{aligned} \tag{4}$$

Here, n_{vb} is v-direction order of $\mathbf{S}_b$,

$$\mathbf{V}_{ai} = \mathbf{P}_{a1i} - \mathbf{P}_{a0i}, \mathbf{V}_{bi} = \mathbf{P}_{a0i} - \mathbf{P}_{b2i}, \mathbf{V}_{ci} = \mathbf{P}_{a0(i+1)} - \mathbf{P}_{a0i}. \tag{5}$$

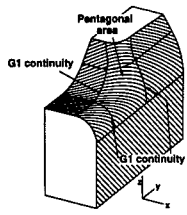


Fig. 3 G^1 connection with surrounding surface.

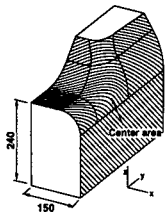


Fig. 4 Shape control result of center area.

By the same procedure, continuity between $\mathbf{S}_a$ and $\mathbf{S}_d$ is corrected to G^1 using quadratic Bézier cross boundary derivatives. $\mathbf{P}_{a14}$ is calculated by the same method to satisfy the twist compatibility. Next, continuity between sub-patches $\mathbf{S}_{a1}$ and $\mathbf{S}_{a2}$ are connected to G^1. Generally, it is difficult to satisfy twist compatibility conditions around the C^0 continuous vertex. Therefore, in order to solve this problem, new control points are generated by knot insertion and these control points are corrected to satisfy the twist compatibility. In this method, boundary curves and cross boundary derivative curves are converted NURBS. $\mathbf{P}'_{a02}$ is generated by knot insertion between $\mathbf{P}_{a02}$ and $\mathbf{P}_{a03}$. Knot vectors are computed by the following procedure:

Case 1: (the number of control points and the order are the same). If a knot of 0.5 is inserted, all the control points of the section will be regenerated by uniform distance. Then, we used 0.5 for the knot vector for both cases; generating a new control points at the start and end of the boundary curve.

Case 2: (piecewise boundary curve). By the same considerations, the knot is set to $knotv[order]/2.0$ for generating new control points at the start of the boundary curve. At the end of the boundary curve, the knot value is set to $1.0 - (1.0 - knotv[cnum - 1])/2.0$. Here, $knotv[]$ is knot vectors. $cnum$ is the number of control points.

These knot values should be adjusted by designer requirements. By the same procedure, $\mathbf{P}'_{a12}$,$\mathbf{P}'_{a22}$,$\mathbf{P}'_{a32}$, $\mathbf{P}'_{a04}$,$\mathbf{P}'_{a14}$,$\mathbf{P}'_{a24}$ and $\mathbf{P}'_{a34}$ can be derived. The control points $\mathbf{P}'_{a14}$ and $\mathbf{P}'_{a24}$ are calculated by the equation (4) to satisfy twist compatibility.

This method can be used to interpolate any number of non-four-sided area (over 4 sided). The G^1 continuity can be satisfied with the surrounding surface, and the twist compatibility condition can also be satisfied.

The surface shape of the boundary area and the center area can be modified smoothly by using the NBG patch shape control method. Fig. 3 shows a surface interpolation result for a pentagonal area. Continuity with neighboring surface is corrected to G^1. Contour curves of the surface are smooth. The degree of $\mathbf{S}^c$ is elevated to quartic and the center control point is moved 30 mm for surface normal direction at $u = 0.5, v = 0.5$. Fig. 4 shows the interpolation result. The center area of the surface can be modified to obtain G^1 continuity with the neighboring surfaces.

2.3.2 Case of piecewise and rational boundary curves

The Konno method [7] is used to join G^1 between piecewise $\mathbf{S}_{a1}$ and piecewise $\mathbf{S}_b$. If the boundary curve is of rational form, the Chiyokura method [3] is used. If the boundary curve is of rational form and the weights of $\mathbf{P}_{a03}$ and $\mathbf{P}'_{a02}$ are different, we should use the LIU method[10] to make a G^1 connection between $\mathbf{S}_{a1}$ and $\mathbf{S}_{a2}$.

§3. Concave Areas

In this section, the surface interpolation result is described for concave areas. Firstly, we apply it to a one-sided concave area. Next, we consider a two-sided concave area.

3.1 Basic idea

It is important to select the four side edges to interpolate the concave area. Our basic idea is that smooth surface have a smooth streamlines. We selected 3 parameters to evaluate the smoothness of streamlines as follows:

1) Max difference of isoparametric line length : MDI
2) Max difference of variation of isoparametric line length : MDVI
3) Max difference of isoparametric line width : MDIW

Here, 1) and 2) evaluate the variation of streamline length and, 3) evaluates the variation of the width of streamline. We believe that the surface can be smooth if these parameters take a small value. Designing four-side edges proceeds as follows:

1) Check concave area,
2) Decide the axis to make a symmetrical shape,
3) Surface edges are selected by symmetry, and concave area curves are composed as one edge,
4) Interpolate by NURBS boundary Gregory patch,
5) Evaluate the surface shape by the evaluation parameter,
6) Make the best edge selection.

3.2 Interpolation for a one-sided concave area

The surface interpolation method is applied to a one-sided concave area. Figs. 5–7 show the results of interpolation for a simple case. The concave area is generated by difference boolean operation to cubic, which is cut by the free form surface. Fig. 8 shows the evaluation results of the surface. Pattern 1 takes small parameter values and isoparametric lines and shading images are smooth as shown in these figures.

Figure 9 shows 3 types of surface interpolation results for deep concave areas which are modeled by the difference boolean operation for a cubic which is cut by free form surfaces. Figure 10 shows the evaluation results using

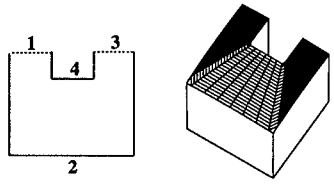


Fig. 5. One-sided concave area: pattern 1

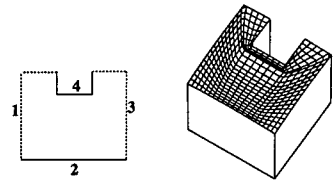


Fig. 6. Pattern 2.

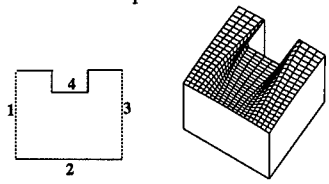


Fig. 7. Surface interpolation of one-sided concave area: pattern 3.

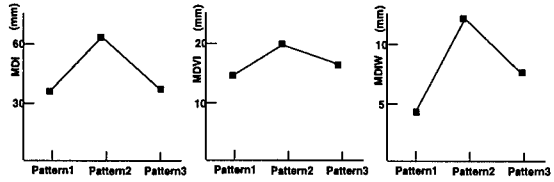


Fig. 8. Surface evaluation result.

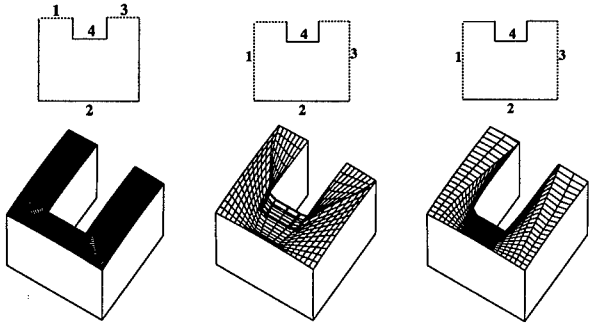


Fig. 9. Surface interpolation of one side concave area: large case.

the previous parameters. Pattern 1 takes small parameter values, and the isoparametric lines are smooth as shown in these figures.

Figure 11 a) shows surface isoparametric lines of complex concave area, b) is the shaded image, and c) shows the control points of S^u. The interpolated surface is smooth as shown in this figure. From these results, smooth surfaces can be generated by selecting the four-side edges to minimize the MDI, MDVI and MDIW. We can get smooth surface for complex concave area.

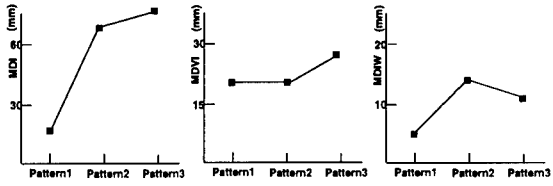


Fig. 10. Surface evaluation result.

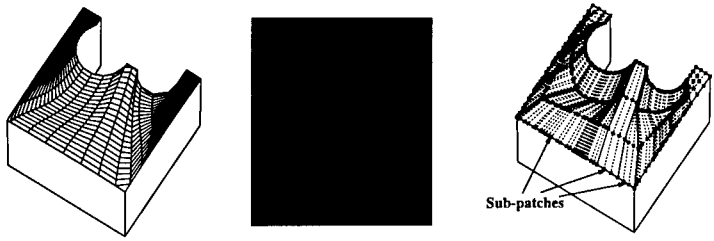


Fig. 11. a) Complex case b) Shading image c) Control points of S^u.

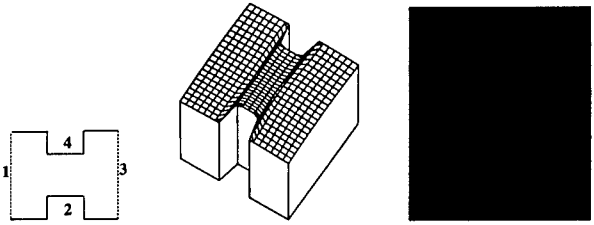


Fig. 12. Surface interpolation of two side concave area : pattern 1.

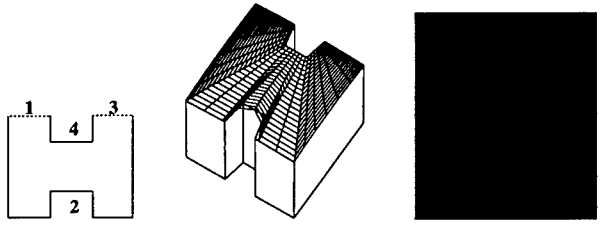


Fig. 13. Surface interpolation of two side concave area : pattern 2.

3.3 Two side of concave area

In the same manner, a simple two-sided concave area is modeled by the difference boolean operation for a cubic which is cut by a free form surface. Figs. 12-14 show the surface interpolation results. Fig. 15 shows the evaluation result for the generated surface. Although pattern 1 shows the lowest strain of the surface from the shaded image, it still has a strain in the middle area. In this case, MDI and MDIW are smaller for the low strain surface.

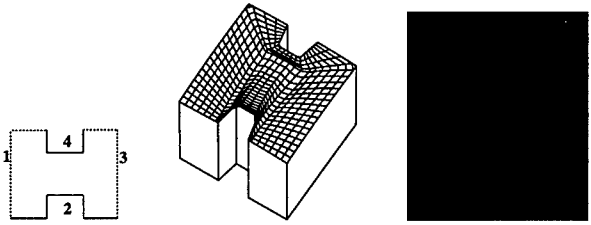


Fig. 14. Surface interpolation of two side concave area : pattern 3.

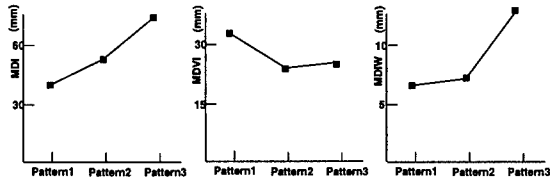


Fig. 15. Surface evaluation result.

From these results, two-sided concave cases require further study.

§4. Conclusion

We propose a non-four-sided interpolation method using a single NBG patch for a concave area. With this method, sub-patches are generated dividing at C^0 continuous point, component surfaces $\mathbf{S}^u$ and $\mathbf{S}^v$ are constructed by merging these sub-patches, which correct continuity.

We selected the 3 parameters to evaluate the smoothness of the surface streamlines. Selecting the four edges to minimize these parameters can generate a smooth surface. A one-sided concave area can be interpolated smoothly for complex cases. For two-sided concave case, evaluation parameter is effective, however, further research is necessary to improve the surface strain.

References

1. Catmull, Edwin and J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, Computer-Aided Design **10(6)** (1978), 350–355.
2. Charrot, P. and J. A. Gregory, A pentagonal surface patch for computer aided geometric design, Comput. Aided Geom. Design **1** (1984), 87–94.
3. Chiyokura, H., T. Takamura, K. Konno, and T. Harada, G^1 surface interpolation over irregular meshes with rational curves, in *Nurbs for Curve and Surface Design,* G. E. Farin (ed), SIAM Publications, Philadelphia, 1991, 15–34.
4. Chiyokura, H. and F. Kimura, Design of solids with free-form surfaces, Computer Graphics **17** (1983), 289–298.

5. Doo, D. W. and M. A. Sabin, Behaviour of recursive division surfaces near extraordinary points, Computer-Aided Design **10(6)** (1978), 356–360.
6. Gregory, J. A., N-sided surface patches, in *The Mathematics of Surfaces,* J.A. Gregory (ed), Clarendon Press, Oxford, 1986, 217–232.
7. Konno, K. and H. Chiyokura, G^1 connection around complicated curve meshes by using C^1 NURBS boundary Gregory patches, Journal of Information Processing Society of Japan **39(1)** (1998), 70–80.
8. Konno, K. and H. Chiyokura, An Approach of Designing and Controlling Free-Form surfaces Using NURBS boundary Gregory patch, Comput. Aided Geom. Design **13(9)** (1996), 825–849.
9. Konno, K., T. Takamura, and H. Chiyokura, A new control method for free-form surfaces with tangent continuity and its applications, in *Scientific Visualization of Physical Phenomena,* N. M. Patrikalakis, (ed), Springer-Verlag, Tokyo, 1991, 435–456.
10. Liu, D., GC^1 continuity condition between two adjacent rational Bézier surface patches, Comput. Aided Geom. Design **7** (1990), 151–163.
11. Sarraga, R., Errata: G^1 interpolation of generally unrestricted cubic Bézier curves, Comput. Aided Geom. Design **6** (1989), 167–171.
12. Sone, J., K. Konno, and H. Chiyokura, Surface interpolation using NURBS boundary Gregory patch for non-four-sided area, Journal of Information Processing Society of Japan **40(2)** (1999), 710–718.
13. Sone, J. and H. Chiyokura, Surface highlight control using quartic blending NURBS boundary Gregory patch, Journal of Information Processing Society of Japan **37(12)** (1996), 2212-2222.
14. Sone, J., T. Watanabe, S. Yamakawa, and H. Chiyokura, NURBS boundary Gregory patch and its shape control, International Journal of Shape Modeling **1(2)** (1994), 151-165.
15. Tiller, Wayne, Rational B-splines for curve and surface representation, IEEE Comp. Graph. Appl. **3(6)** (1983), 61–69.

J. Sone
Toshiba Corporation, 1 Toshiba-tyou, Futyuu, Tokyo, 183-0043, Japan
junji.sone@toshiba.co.jp

K. Konno
Ricoh Corporation, 1-1-17 Koishikawa, Bunkyou, Tokyo, 112-0002, Japan
konno@src.ricoh.co.jp

H. Chiyokura
Faculty if Environmental Information, Keio University,
5322 Endo, Fujisawa, Kanagawa, 252-0816, Japan
chiyo@sfc.keio.ac.jp

Central Conics on Parabolic Dupin Cyclides

Kenji Ueda

Abstract. Hyperbolas, ellipses and degenerate conics on parabolic Dupin cyclides are investigated. These central conics are obtained as the intersections of parabolic cyclides and the planes perpendicular to the two planes of symmetry of the cyclides. They are also the images of central conics in the parametric space. Since the conics are planar curves, they are transformed into planar or spherical curves on Dupin cyclides via inversion. Lemniscates of Bernoulli on Dupin cyclides and Viviani's curves on right-circular cylinders are included in the inverted conics. Two intersecting lines on a parabolic ring cyclide, which are degenerate conics, are inverted into Villarceau circles on a ring cyclides.

§1. Introduction

It is known that any Dupin cyclide [1,2,5,6], which is a quartic surface is the image under inversion of a torus, circular cylinder or circular cone. Parabolic cyclides, which are cubic surfaces, are obtained as an inverse surface with its inversion center at a point on the Dupin cyclides. While quartic cyclides are closed surfaces, cubic cyclides are unbounded in extent.

Since both of them are expressed as biquadratic rational surfaces [4,5,6,7], the image of a rational curve on the parameter space of a Dupin cyclide is a rational curve on the cyclide. It is well known that the isoparametric curves on a Dupin cyclide, which are lines of curvature of the cyclides, are circular.

In this paper, non-circular rational curves of lower degree on Dupin cyclides, especially central conics on parabolic cyclides, are investigated. The other curves are derived from conics on cyclides via inversion.

§2. Conics on Parabolic Dupin Cyclides

Let the curves $c_1(u)$ and $c_2(v)$ be the focal parabolas given by

$$c_1(u) = \frac{1}{2}\begin{pmatrix} q+(p-q)u^2 \\ 2(p-q)u \\ 0 \end{pmatrix}, \qquad c_2(v) = \frac{1}{2}\begin{pmatrix} p+(q-p)v^2 \\ 0 \\ 2(q-p)v \end{pmatrix}, \qquad (1)$$

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 399–406.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

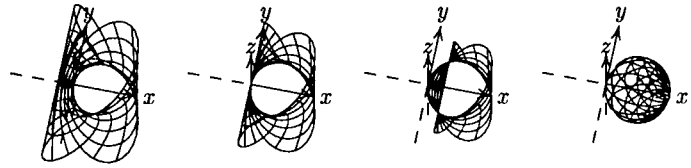


Fig. 1. Parabolic cyclides and a sphere.

and the functions $r_1(u)$ and $r_2(v)$ be the radius functions given by

$$r_1(u) = \frac{q - (p-q)u^2}{2}, \qquad r_2(v) = \frac{p - (q-p)v^2}{2}. \tag{2}$$

Parabolic cyclides are parameterized with the parameters u and v and the shape parameters p and q as [4,6]

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \frac{r_1(u)c_2(v) - r_2(v)c_1(u)}{r_1(u) - r_2(v)} = \frac{1}{u^2+v^2+1} \begin{pmatrix} pu^2 + qv^2 \\ (p-(q-p)v^2)u \\ (q-(p-q)u^2)v \end{pmatrix}. \tag{3}$$

The isoparametric curves, which are circles, are the curvature lines of the parametric surface. Since the radius functions $r_1(u)$ and $r_2(v)$ represent the curvature radius, the principal curvatures κ_1 and κ_2 are their reciprocals, i.e., $\kappa_1 = 1/r_1(u)$ and $\kappa_2 = 1/r_2(v)$.

By eliminating the parameters u and v, we obtain the following implicit form of the parabolic cyclide (3):

$$(x-q)y^2 + (x-p)z^2 + x(x-p)(x-q) = 0. \tag{4}$$

Equation (4) implies that the cross section between the surface and the plane parallel to the yz-plane is a central conic.

From the x-value of (3), we obtain another central conic in the parameter space of the surface:

$$(x-p)u^2 + (x-q)v^2 + x = 0. \tag{5}$$

The central conics of (4) on parabolic cyclides are the images of the central conics of (5) in the parametric space.

The x-value of (3) is also transformed into

$$x = \frac{u^2}{u^2+v^2+1}p + \frac{v^2}{u^2+v^2+1}q + \frac{1}{u^2+v^2+1}0. \tag{6}$$

Hence, the x-value is the convex combination of p, q and 0. From the symmetry of (3), we can restrict the domain of the shape parameters p and q within $0 < p$ and $q \leq p$ for the classification of the shape of the surface. Since one of the principal curvatures is always positive, i.e., $0 \leq \kappa_2$, under this restriction, the Gaussian curvature $K = \kappa_1\kappa_2$ has the same sign as that of another principal curvature κ_1. Since parabolic cyclides extend to infinity, Figure 1 illustrates several shapes of the parabolic cyclide for $(u,v) \in ([-\infty,\infty] \times [-1,1]) \cup ([-1,1] \times [-\infty,\infty])$.

2.1 Hyperbolas on parabolic ring cyclides $(q < 0 < p)$

x	(y,z)	κ_1	(u,v)
$x = q$	$z = \pm 0$	$\kappa_1 < 0$	$v = \pm\infty$
$q < x < 0$	$\dfrac{z^2}{-x(x-q)} - \dfrac{y^2}{-x(p-x)} = 1$	$\kappa_1 < 0$	$\dfrac{v^2}{\frac{-x}{x-q}} - \dfrac{u^2}{\frac{-x}{p-x}} = 1$
$x = 0$	$z = \pm\sqrt{\dfrac{-q}{p}}\,y$	$\kappa_1 < 0$	$v = \pm\sqrt{\dfrac{p}{-q}}\,u$
$0 < x < p$	$\dfrac{y^2}{x(p-x)} - \dfrac{z^2}{x(x-q)} = 1$	$\kappa_1 < 0$	$\dfrac{u^2}{\frac{x}{p-x}} - \dfrac{v^2}{\frac{x}{x-q}} = 1$
$x = p$	$y = \pm 0$	$\kappa_1 = 0$	$u = \pm\infty$

There are two straight lines at $x = 0$ on a parabolic ring cyclide.

2.2 Hyperbolas on thorn cyclides $(q = 0 < p)$

x	(y,z)	κ_1	(u,v)
$x = 0$	$z = \pm 0$	$\kappa_1 \leq 0$	$u = 0$ or $v = \pm\infty$
$0 < x < p$	$\dfrac{y^2}{x(p-x)} - \dfrac{z^2}{x^2} = 1$	$\kappa_1 < 0$	$\dfrac{u^2}{\frac{x}{p-x}} - v^2 = 1$
$x = p$	$y = \pm 0$	$\kappa_1 < 0$	$u = \pm\infty$

There is a singular point at the origin on a thorn cyclide.

2.3 Ellipses and hyperbolas on parabolic horn cyclides $(0 < q < p)$

x	(y,z)	κ_1	(u,v)
$x = 0$	$y = z = 0$	$\kappa_1 > 0$	$u = v = 0$
$0 < x < q$	$\dfrac{y^2}{x(p-x)} + \dfrac{z^2}{x(q-x)} = 1$	$\kappa_1 > 0$	$\dfrac{v^2}{\frac{x}{q-x}} + \dfrac{u^2}{\frac{x}{p-x}} = 1$
$x = q$	$z = \pm 0$	$\kappa_1 = \infty$	$u = \pm\sqrt{\dfrac{q}{p-q}}$
$q < x < p$	$\dfrac{y^2}{x(p-x)} - \dfrac{z^2}{x(x-q)} = 1$	$\kappa_1 < 0$	$\dfrac{u^2}{\frac{x}{p-x}} - \dfrac{v^2}{\frac{x}{x-q}} = 1$
$x = p$	$y = \pm 0$	$\kappa_1 = 0$	$u = \pm\infty$

There are two singular points $(q, \pm\sqrt{(p-q)q}, 0)$ on a parabolic horn cyclide.

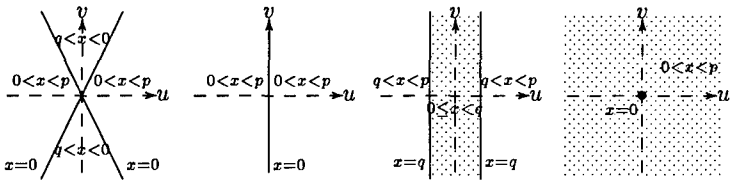


Fig. 2. Parameter space.

2.4 Circles on spheres $(0 < q = p)$

x	(y, z)	κ_1	(u, v)
$x = 0$	$y = z = 0$	$\kappa_1 = \frac{2}{p}$	$u = v = 0$
$0 < x < p$	$y^2 + z^2 = x(p - x)$	$\kappa_1 = \frac{2}{p}$	$u^2 + v^2 = \frac{x}{p - x}$
$x = p$	$y = z = 0$	$\kappa_1 = \frac{2}{p}$	$u = \pm\infty$ or $v = \pm\infty$

The implicit form (4) of parabolic cyclides becomes the quadratic form of $y^2 + z^2 + x(x - p) = 0$.

2.5 Central conics on parabolic cyclides

On cubic cyclides, there are hyperbolas on the region with negative Gaussian curvature, and ellipses on the region with positive Gaussian curvature. Since the central conics are the images of central conics in the parameter space, the parameter space is subdivided by straight lines into regions corresponding to the geometric properties of their images, as illustrated in Figure 2. The preimages of the singular points are the straight lines parallel to the v-axis in the parameter space. The shaded regions are the preimages of the surface regions with positive Gaussian curvature.

For $0 < x < p$, the centers of the central conics are at $(x, 0, 0)$, the vertices are at the points $(x, \pm\sqrt{(p - x)x}, 0)$, and the foci are at the points $(x, \pm\sqrt{(p - q)x}, 0)$. The foci lie on the parabola $y^2 = (p - q)x$.

The central conics at $x = (p + q)/2$ are rectangular hyperbolas

$$y^2 - z^2 = \frac{p^2 - q^2}{4}, \qquad \left(u^2 - v^2 = \frac{p + q}{p - q}\right). \tag{7}$$

The mean curvature $H = (\kappa_1 + \kappa_2)/2$ is zero at any point along the rectangular hyperbolas on parabolic cyclides.

§3. Inverse Surfaces of Parabolic Cyclides

The inverse surfaces of the parabolic cyclides with inversion radius 1 with respect to a point $(r, 0, 0)$, which is the center of the central conics, are expressed

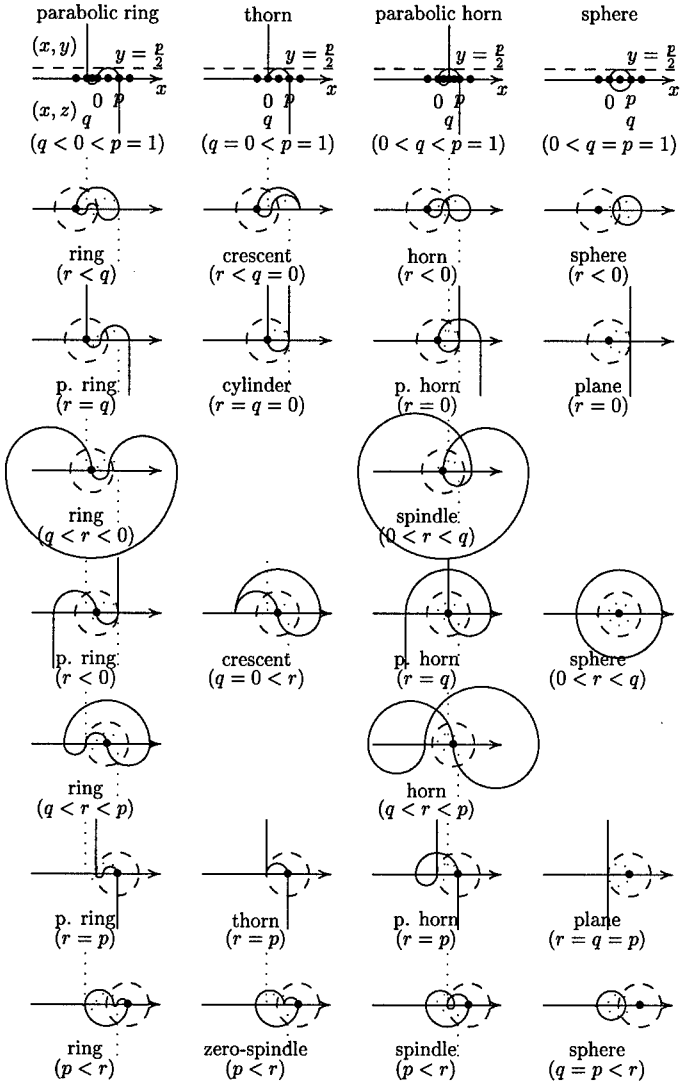


Fig. 3. Inversion of parabolic cyclides.

as

$$
\begin{aligned}
X &= \frac{x-r}{(x-r)^2+y^2+z^2} + r = \frac{(p-r)u^2+(q-r)v^2-r}{(p-r)^2u^2+(p-q)^2u^2v^2+(q-r)^2v^2+r^2} + r,\\
Y &= \frac{y}{(x-r)^2+y^2+z^2} = \frac{(p-q)v^2+p}{(p-r)^2u^2+(p-q)^2u^2v^2+(q-r)^2v^2+r^2}u,\\
Z &= \frac{z}{(x-r)^2+y^2+z^2} = \frac{(q-p)u^2+q}{(p-r)^2u^2+(p-q)^2u^2v^2+(q-r)^2v^2+r^2}v.
\end{aligned}
\tag{8}
$$

Since the inversion center $(r,0,0)$ is on both planes of symmetry, the xy- and xz-planes, the inverse surfaces have the same planes of symmetry. The inverse surfaces are illustrated in Figure 3 with the cross sections between the surfaces and their planes of symmetry. The points are the inversion centers and the dashed curves are the inversion spheres in the figure. Equation (8) is a parameterization of Dupin cyclides with the shape parameters p, q and r.

The two intersecting lines on parabolic ring cyclides are inverted to two circles on ring cyclides. Since the circles are called Villarceau circles, we may call the two straight lines Villarceau lines.

While right circular cylinders are obtained as the inverse surfaces of thorn cyclides with the inversion center at the origin, as illustrated in Figure 3, right circular cones are obtained via inversion of parabolic horn cyclides in a circle with the center at one of the pinch points $(q, \pm\sqrt{(p-q)q}, 0)$. The resultant circular cones are obtained with a half-vertex angle of $\arctan\sqrt{q/(p-q)}$.

Various cyclides are obtained via inversion by specifying various points as the inversion center.

§4. Rational Curves on Dupin Cyclides

Special rational curves [3] on Dupin cyclides are shown in this section. Since the curves are inverse curves of conics, they are planar or spherical. Figure 3 will be useful in imaging the rational curves. The rational curves include the inverse curves of conics on right circular cones, because the inverse surfaces of right circular cones are also Dupin cyclides.

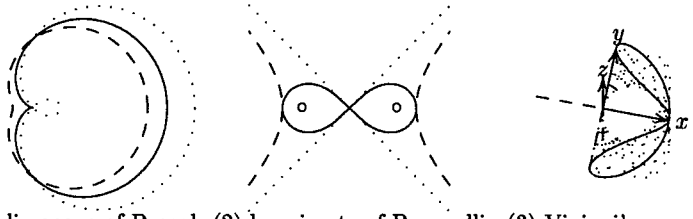
4.1 Rational quartics on Dupin cyclides

The polar equation of a conic, of which the focus is at the pole, is given by $r = l/(1 + e\cos\theta)$, where e is the eccentricity and l is a constant. The conic is a parabola ($e = 1$), an ellipse ($0 \leq e < 1$), a circle ($0 = e$), a hyperbola ($1 < e$) or a rectangular hyperbola ($e = \sqrt{2}$).

The inverse curve $r = (1 + e\cos\theta)/l$ is called a limaçon of Pascal, and is expressed as a rational quartic. Hence, there are limaçons of Pascal on Dupin cyclides as the inverse curves of a conic with the inversion center at their foci. Some limaçons of Pascal are illustrated in Figure 4 (1). In the case of $e = 1$, a limaçon of Pascal is called a cardioid. There are cardioids on horn cyclides inverted from a circular cone with its inversion center at the focus of a parabola on the cone.

The inverse of rectangular hyperbolas $x^2 - y^2 = a^2$ with respect to the center are obtained as the curves $a^2(x^2+y^2)^2 = x^2 - y^2$. The curves are called lemniscates of Bernoulli, and are illustrated in Figure 4 (2). The lemniscates are also rational quartic curves on Dupin cyclides.

The inverse surface of the thorn cyclide in the sphere $x^2 + y^2 + z^2 = p^2$ is the right circular cylinder $(x - p/2)^2 + z^2 = (p/2)^2$, as illustrated in Figure 3. The intersection of the sphere and the cylinder is called a Viviani's curve. As the intersection is fixed under the inversion, the Viviani's curve is also on the



(1) limaçons of Pascal (2) lemniscate of Bernoulli (3) Viviani's curve

Fig. 4. Rational quartics on Dupin cyclides.

thorn cyclide, and is expressed as

$$\frac{p}{(1+t^2)^2}\begin{pmatrix}(1-t^2)^2\\2t(1+t^2)\\-2t(1-t^2)\end{pmatrix}, \tag{9}$$

which is the image of the curve in the parameter space

$$u^2-\frac{1}{v^2}=1, \qquad (u,v)=\left(\frac{1+t^2}{2t},\frac{2t}{1-t^2}\right). \tag{10}$$

Figure 4 (3) illustrates the Viviani's curve. The Viviani's curve is inverted into a rectangular hyperbola with the inversion center at the knot of the curve.

4.2 Rational cubics on parabolic cyclides

The intersection between a parabolic cyclide and the plane $y = p/2$ is expressed as $(x-p)z^2+(x-q)(x-p/2)^2 = 0$ from (4). The curve is parameterized with a rational cubic as

$$x(t)=\frac{q+pt^2}{1+t^2}, \qquad z(t)=\frac{t(q-\frac{p}{2}(1-t^2))}{1+t^2}. \tag{11}$$

The curve is obtained as the pedal curve [3] of the parabola $z^2 = -2(x-q)$ with respect to the point $(x,z) = (p/2,0)$. A pedal curve is the locus of the foot of the perpendicular from a fixed point to a variable tangent to a given curve. The curve has an asymptote $x = p$ and a singular point (node, cusp or isolated point) at $(p/2,0)$. The curve may be called a trisectrix of Maclaurin ($q = -p$), a right strophoid ($q = 0$), or a cissoid of Diocles ($q = p/2$). Figure 5 illustrates the curve for various values of the shape parameter q.

The inverse curve in the circle $(x-p/2)^2+z^2 = 1$ becomes

$$X=\frac{x-\frac{p}{2}}{(x-\frac{p}{2})^2+z^2}=\frac{1}{q+\frac{p}{2}(t^2-1)}, \quad Z=\frac{z}{(x-\frac{p}{2})^2+z^2}=\frac{t}{q+\frac{p}{2}(t^2-1)}, \tag{12}$$

and is implicitized into the conic $(p-2q)X^2+2X-pZ^2 = 0$. Therefore, the cubic curve is an inverse curve of a conic on a Dupin cyclide.

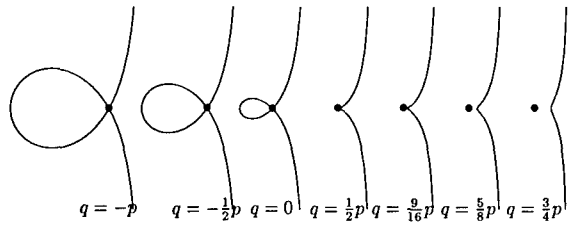


Fig. 5. Rational cubics on parabolic cyclides.

§5. Conclusion

It has been shown that there are central conics on parabolic Dupin cyclides. The central conics are cross sections between parabolic cyclides and the plane perpendicular to the two planes of symmetry of the cyclides. The preimage curves in the parameter space are also central conics.

Special cubic or quartic curves, which have been investigated for a long time, are also found on Dupin cyclides. They are planar or spherical rational curves and the images of conics on Dupin cyclides under inversion.

References

1. Albrecht, G. and W. L. F. Degen, Construction of Bézier rectangles and triangles on the symmetric Dupin horn cyclide by means of inversion, Comput. Aided Geom. Design **14** (1997), 349–357.
2. Chandru, V., D. Dutta, and C. M. Hoffmann, On the geometry of Dupin cyclides, The Visual Computer **5** (1989), 227-290.
3. Lawrence, L. D., *A Catalog of Special Plane Curves*, Dover, 1972.
4. Peternell, M. and H. Pottmann, Designing rational surfaces with rational offsets, *Advanced Topics in Multivariate Approximation*, F. Fontanella, K. Jetter and P. J. Laurent (eds.), World Scientific, 1996, 275–286.
5. Pratt, M. J., Cyclides in computer aided geometric design, Comput. Aided Geom. Design **7** (1990), 221–242.
6. Pratt, M. J., Cyclides in computer aided geometric design II, Comput. Aided Geom. Design **12** (1995), 131–152.
7. Srinivas, Y. L. and D. Dutta, Rational parametric representation of parabolic cyclide: Formulation and applications, Comput. Aided Geom. Design **12** (1995), 551–566.

Kenji Ueda
Ricoh Company, Ltd.
1-1-17, Koishikawa, Bunkyo-ku, Tokyo 112-0002, Japan
ueda@src.ricoh.co.jp

Adaptive Parameterization and Approximation for CAD Data Reduction

G. Wahu, J. M. Brun, and A. Bouras

Abstract. Data reduction is frequently needed in curve and surface model conversion, primarily for CAD data exchanges, but also to ease the designer's work. Such data reduction corresponds globally to a number of pole reductions. A previous analysis has shown that there are two main criteria in this domain: the parametrization and the extremity conditions. The possibility of defining an optimum optimorum solution for the approximations used in data reduction is noted. The relation between this optimum optimorum and the existence of an optimal parametrization leads to a new approach for curve and surface approximation. This approach has the advantage of modifying the parameter settings in a transparent way, while matching easily the extremity conditions. Finally, the extension to surface data reduction of this scheme is presented.

§1. Introduction

Designers have to create shapes in highly constrained environments. Curves and surfaces have to join precisely under tangency and curvature conditions; they have to meet some points as precisely as possible while behaving "nicely" between these points. All these conditions result into curves and surfaces either over-segmented or of dangerously high degree, and eventually both. Such curves and surfaces that are defined by a number of poles larger than necessary, induce severe problems in further use. Data exchange with other systems can be impossible if the degree exceeds what is allowed in the receiving system, or untractable if the number of curves or surfaces generated is too large. Data quality can be very poor. Curves and surfaces which are defined by an extremely large number of poles have difficulties in behaving "nicely". One knows also that the parametrization of curves and surfaces is crucial when defining them by discrete points [7,9,10,12]. Strangely, it is generally considered that CAD data reduction must be done with the original parametrization.

One considers also that the quality of a data reduction is directly related to the distance between points of a same parameter value, doing so

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 407–416.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

one mixes distances perpendicular to the original curve with tangential distances corresponding to different parametrizations. One must remark also that some approaches cannot meet essential conditions such as end conditions. We present here some results of our search for the optimal parametrization which minimizes the amount of data needed to fit curves and surfaces to a given precision.

§2. Previous Approaches

2.1 Degree reduction of Bézier curves

This problem was addressed early and is considered as the inverse problem to degree elevation [3,4,5,6,7,11,14]. Degree elevation is obtained step by step by a de Casteljau process of poles creation, which can be done either from poles π_0 to π_n or π_n to π_0 which produces two different results depending on the way it is processed. The idea of a blend between these two elementary processes, with blending coefficients depending on the rank in each elementary process, is a natural one. It was an important insight in this process that enabled Eck [5,6] to find the optimal blending and to prove that it is the best componentwise approximation. However, this optimal solution, aside from its componentwise limitation, assumes that the curve parametrization stays unchanged. Improvements over the Eck's solution can be expected from a global optimization and parametrization modifications.

2.2 Reduction in the number of poles for B-splines

The reduction of the number of poles is a crucial problem when converting from B-Splines to the Bézier form. B-Splines can be decomposed into Bézier curves or patches of low degree (typically 3 with C^2 continuity). Doing so produces huge sets of low degree Bézier curves or patches, which are unmanageable on most Bézier based CAD systems.

2.3 Classical parametrization schemes

Classical parametrization schemes were presented by Farin [7]. His conclusions are that it is good practice to test chordal parametrization first, then Lee's centripetal scheme and ultimately Foley's tangent variations. Such parametrizations come from kinematic analogies where one travels on the curve at constant speed or slows down on curves depending on centrifugal forces or the speed of turning the steering wheel. These schemes, while grounded in common sense, were probably found too empiric by Hosheck [10], who proposed a scheme relating parameter modifications to tangential errors. In this scheme, for a given parametrization, a least square minimization produces a curve minimizing both normal and tangential errors. Iterative modifications of the parametrization using, at each step, the parametrization produced by the preceding least square modifications would be extremely computer intensive. Hosheck uses instead a projection of the errors on the curve tangents for each parameter value, in order to optimize the parametrization. However, the

process remains computer intensive since it still has to loop on least square minimizations. Moreover the process can have convergency difficulties that can stop the process far from it's goal. This goal is to reach a parametrization producing normal errors, called "intrinsic parametrization" by Hosheck. The optimal parametrization sought here can be defined for the approximation of curves or surfaces known everywhere, or nearly everywhere. It is the parametrization for which the maximum distances between each point on a curve and the approximated curve is minimal, which implies that these distances are perpendicular to the curve.

§3. Proposed Approach

3.1 Early approaches to optimal parametrization

In his thesis, P. Bézier [1] considers implicitly that a parametrization proportional to curvilinear abscissa is optimal, and that chordal parametrization of sparse sets of points is an approximation that can be slightly improved taking circles passing through points taken 3 by 3. At the same time (1975) J. M. Brun, in a non published study, took the Bézier curve defined by the sparse set of points, taken as poles of the curve, and used their projections on this curve as parameters for the points. He found that improvements over this parametrization are possible when an iterative modification of the curve is made using normal errors to move poles and tangential errors to modify the parametrization. The computing power available by that time imposed to stop the process after a small number of iterations, and the scarce set of points prevented to define what can be an optimal parametrization, since the definition of a distance between curves was not possible.

3.2 Our search of an optimal parametrization

Following similar goals as Hosheck [10], we have tried to define intrinsic parametrization of curves, related only to shape characteristics. The first characteristic of a curve is it's length, corresponding to the curvilinear abscissa s as an intrinsic parametrization of the curve. Then, any other parametrization can be defined as a parametrization law: $t = f(s)$. We found that a mathematically sound strategy can be to approximate the optimal law $t = f(s)$ by a power series $t = s(a + b * s + c * s^2 + \cdots)$ [2]. The coefficients of this series can be derived from geometric extremity conditions: end points produce $t = s$ (called a linear law), end tangents produce $t = s(a + b * s)$ (called a parabolic law) and adding curvatures produce $t = s(a + b * s + c * s^2)$ (called a cubic law). Depending on the curve shapes, the law to use needs more or less completion, and the results are convincing up to curve shapes with one inflexion. For curve shapes of higher complexity, higher degree series are needed, and the cubic scheme was found complex enough to avoid going further. In such cases, a segmentation of the curve allowing a piecewise approximation of the $t = f(s)$ law was envisioned.

3.3 Are designer's skills inherently better than mathematics?

While experimenting with our mathematical approaches, either improving the degree of $t = f(s)$ or taking care of curvature extrema, it was found that an experienced designer was always able to improve the result [13]. He just moves poles interactively. The parametrization of the designer's curve produces a curve significantly closer to the target curve than any mathematical approach. It was thus considered of interest to analyze the designer's actions, and to reproduce them by appropriate heuristics.

3.4 The designer's algorithm

The experienced designer knows the influence of a pole displacement and "integrates" the errors on the curve (perpendicular to the target curve) as "demands" to modify each pole with "weights" implicitly given by the coefficients $B_i^n(t_j)$, heavily influenced by his experience and adjusted by the feedback of the curve response to his modifications. In this process of curve adjustment by displacing poles, he simultaneously modifies the curve parametrization and the curve shape until he obtains a global optimum for which any pole displacement would increase normal errors at some places more than it would reduce it at others. The experienced designer knows pretty well that a pole displacement has no influence on the normal errors at places parallel to that displacement. In doing so, he has the ability to reduce errors by parametrization improvements, even though he is generally unaware of it. This analysis of the rationale behind the designer's heuristics produces an algorithm that can be called the designer's algorithm:

Extract the sample points P_j from the original curve:

The number of points P_j has a direct influence on the performance of the process. This sample of points can be refined to improve the precision of the approximation and to obtain better convergence.

Construct an initialization curve.

Iterate while the process converges until the desired precision is reached:

1) *For each point P_j, seek for the point $C(t_j)$ of the approximation curve whose normal passes through P_j,*
2) *Check the precision of the approximation,*
3) *Find the maximum errors,*
4) *For each pole, calculate and apply a "displacement demand",*
5) *Study the convergence of the process.*

The "displacement demand" on a pole is that $\delta\pi_i = \sum_j (P_j - C(t_j)) B_i^n(t_j)$, see Fig. 1.

3.5 The math behind the 'designer's algorithm'

When the designer's algorithm reaches it's goal, all the errors are normal to the approximating curve, and the parametrization of the given curve is given

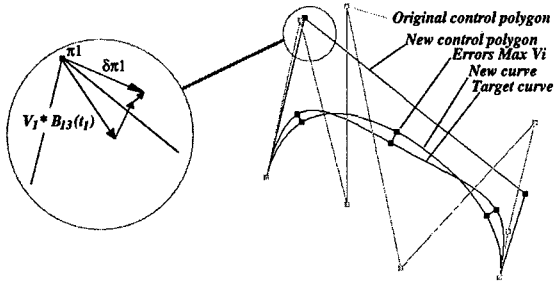


Fig. 1. The designer's algorithm.

by the approximation curve $C(t)$. With such a parametrization, the least square approach minimizes the error function:

$$E = \sum_{j=1}^{m} (P_j - C(t_j))^2,$$

where P_j are again the sampling points, and the approximation is

$$C(t_j) = \sum_{i=0}^{n} \pi_i B_i^n(t_j).$$

The minimizing conditions $\frac{\delta E}{\delta \pi_i} = 0$ leads to

$$\sum_{j=1}^{m} (P_j - \sum_{k=0}^{n} \pi_k B_k^n(t_j)) B_i^n(t_j) = 0; i \in [0, n].$$

For G^0 conditions, $\pi_0 = P_1$ and $\pi_n = P_m$ are the curve's extremities. For G^1 conditions, π_1 and π_{n-1} are constrained to lay on the extremity tangents. For convergence, we can identify the following least square conditions on the designer's algorithm:

$$\sum_{j=1}^{m} (P_j - C(t_j)) B_i^n(t_j) = 0; i \in [0, n].$$

After convergence, the designer's algorithm produces

- a parametrization where errors are normal to the curve,
- the least square solution for this parametrization.

The existence and uniqueness of this solution relies on the convergence of the process and the uniqueness of the parametrization. Aside from proofs of the convergence that mathematicians may provide, we can say that the

convergence of the process is typically slow but regular. However, single iterations are much faster than the mean square resolutions used by Hosheck. Globally, the performance can be compared favorably with Hosheck's intrinsic parametrization, which has the same goal of normal errors but a convergence that can be irregular.

Using a mastering of the process convergence [8], we were able to improve considerably the convergence rate of a regular convergence such as for the designer's algorithm. Depending on the initializing curve, the process can fall in a "convergence trap". To get out such traps, one can envisage using simulated annealing techniques. However, we have observed that our convergence mastering techniques combine jumping over traps to process acceleration. The uniqueness of the parametrization would be ensured by errors normal to the given curve, but it can be questionable for errors normal to the approximating curve. Aside from mathematical proofs, we remark that when the approximation is good, both normals are equivalent and so are the parametrizations. Moreover, if one minimizes the maximum errors only, as with the Tchebychev minimax approach, the normals at these points are common to both curves. This corresponds to a real designer's actions who takes care of maximum errors only, so the designer's algorithm has to use a "displacement demand":

$$\sum_{k=1}^{l} \overrightarrow{\delta_i k}, \qquad i \in [0, n],$$

$$\overrightarrow{\delta_i k} = maxk((P_j - C(t_j))B_i^n(t_j)).$$

The $maxk$ function produces the maximum value on the interval k between two crossings of P_j and $C(t_j)$. This displacement demand corresponds to a curve's distance which is a mix of the least square distance and the minimax of Tchebychev. When using this distance, one speeds up the algorithm and produces a better accuracy since the result is closer to an equioscillating approximation. An extension of the 'designer's algorithm' to B-Splines or NURBS would be straight forward: one has just to replace the $B_i^n(t)$ weighting function by corresponding ones. However, node sequence modifications are not done implicitly by such extensions, and an effective designer's algorithm must include node sequence definition. The extension to surfaces is also straight forward, but it has the additional advantage of working where designers may have trouble moving poles interactively. This extension is the following:

1) Reduce boundary curves complexity and adjust degree on opposite curves.
2) Use a Coons bilinear interpolant of the boundary curves as starting point.
3) Iterate on poles' displacements like for curves.

Pole displacements are modified by the use of errors normal to the approximating surface $S(u, v)$ and weighting functions $B_i^n(u)$ $B_j^m(v)$ in place of $C(t)$ and $B_i^n(t)$. Then

$$\overrightarrow{\delta\pi_{ij}} = \sum_{i=0}^{n}\sum_{j=0}^{m}(P_{kl} - S(u_k, v_l))B_i^n(u_k)B_j^m(v_l); i \in [0, n], j \in [0, m],$$

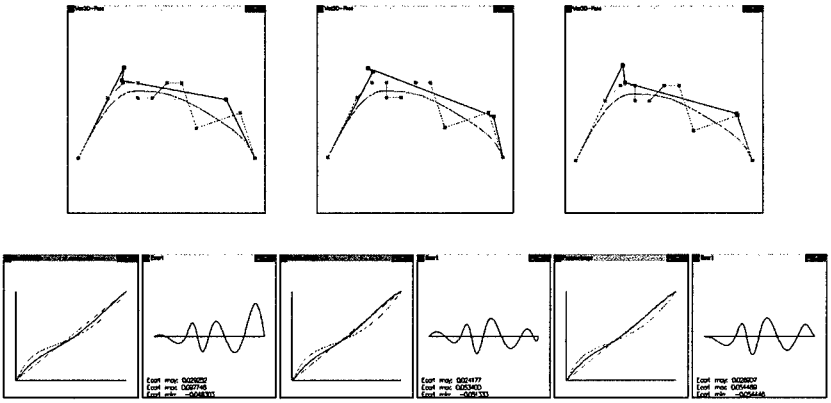


Fig. 2. (a) G^0 Hoschek, (b)–(c): G^0 and C^1 designer's algorithm.

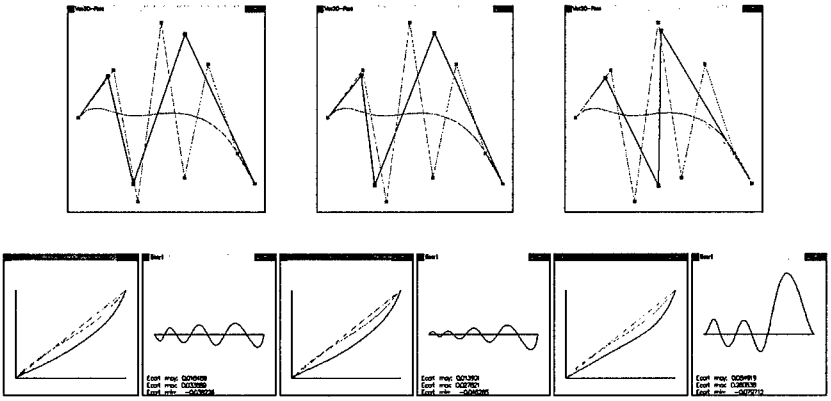


Fig. 3. (a) G^0 Hoschek, (b)–(c): G^0 and C^1 designer's algorithm.

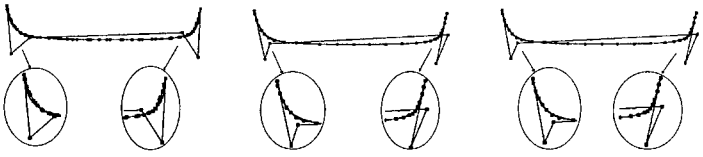


Fig. 4. (a) G^0 Hoschek, (b)–(c): G^0 and C^1 designer's algorithm.

where $S(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \pi_{ij} B_i^n(u) B_j^m(v)$. The 'designer's algorithm' on surfaces has the same behavior as for curves, but the computing time is obviously higher and the convergence mastering much more important. Replacing the least square distance by the local maxima distance would speed up the process quite efficiently like for curves.

§4. Results

4.1 Bézier and B-spline curves

The first test uses a degree-10 Bézier curve, proposed by Bogacki [3]. The first approximations are calculated by using the Hoschek's G^0 solution (Fig.2a). The next curves are the G^0 (Fig.2b) and G^1 (Fig.2c) approximations obtained after application of the proposed 'designer's algorithm'. The precision are respectively: 0.029, 0.024, 0.027. Note our process adapts nicely to the G^1 condition.

The second test curve (Fig. 3) is a 'real' Bézier curve, found in CAD data transfer. Designers feel comfortable in reducing from degree 7 to 4 at the cost of the cancellation of a small inflexion close to the right extremity. Here the smoothed inflexion degrades the designer's algorithm curve when a G^1 condition is needed. These curves precision are respectively 0.018 for Hoschek's G^0 curves, and 0.013, 0.084 for the G^0 and G^1 designer's algorithm curves.

The last test curve (Fig. 4) is also a 'real' curve. It is a B-Spline curve defined by 28 poles with a parametrization law more or less linear. This curve has a somewhat chaotic curvature repartition that can be smoothed vigorously with a degree 5 Bézier curve. The designer's algorithm process again gives much better results than Hoschek's solution with the G^0 condition. The precision obtained is 0.00138 for Hoschek's G^0 curve, and 0.00095 and 0.00097 for our G^0 and G^1 curves, respectively. Since we share with Hosheck the principal of searching a parametrization producing normal errors, our better result is due to better and more regular convergence.

4.2 Surfaces

Surfaces are computationally much more expensive than curves, but the possible data reduction is also much higher. Consider for example a B-spline surface defined by a network of $23 * 23$ poles (Fig. 5a).

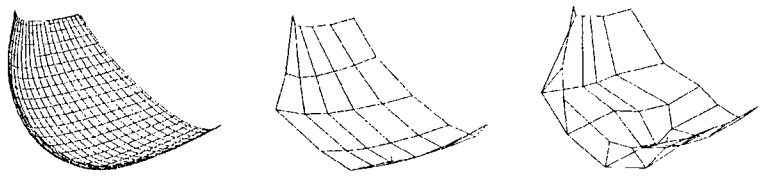


Fig. 5. Control polygons: Original (a) Coons/Bézier (b) 'Designer's Algorithm' (c).

Since it is a $3 * 3$ degree B-spline surface, the number of poles would be $16*22*22$, if converted to a set of $3*3$ degree Bézier patches. It is possible to convert it to a 5*5 degree Bézier surface of 36 poles only. We use the degree-5*5 Coons/Bézier surface (Fig. 5b) as initial surface. Figure (Fig. 6a) represents the evolution of the error along the surface. At this stage of the process, the approximation's relative error is 0.0475. Then the designer's algorithm is

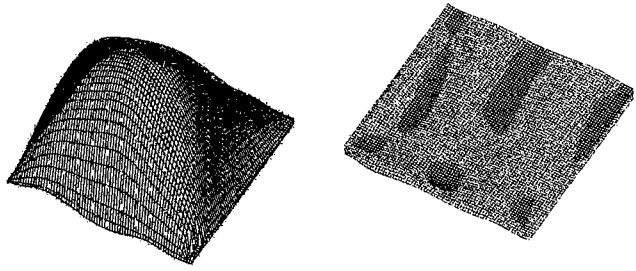


Fig. 6. Errors: Coons/Bézier (a) 'Designer's Algorithm' (b).

applied to the surface (Fig. 5b), and one obtains the surface (Fig. 5c) whose relative error is only 0.00126.

§5. Conclusion

Reducing the number of poles is an important problem for converting curve and surface models, and is needed in CAD data exchanges. Reducing step by step the degree of a Bézier curve shows that the original parametrization is less and less optimal. More generally, the approximation of a dense set of points needs a parametrization adapted to the degree of the approximating curve while related also to the curve shape. An optimal parametrization can be called more adaptive than intrinsic since it has this combined dependency.

One can observe that designers, by moving interactively poles, have the ability to improve easily over most of the mathematical approaches. Doing so, they produce implicitly the needed adaptive parametrization. An analysis of the designer's actions leads to the definition of an algorithm called "the designer's algorithm". This algorithm produces a parametrization with errors normal to the curve, which is adapted both to the curve shape and the degree of the approximation. A simple mathematical analysis shows that depending on the computation of the poles displacements in the designer's algorithm, the result can minimize the least square distance or a local maxima distance similar to the Tchebychev minimax. With the local maxima distance, the result is nearly equioscillating and improves over the least square distance. The designer's algorithm takes into account easily the extremity conditions needed for CAD data, and extends in a straight forward way to surface approximation.

Our algorithm seems to present practical advantages that can justify a more elaborate mathematical analysis than presented here. Mathematicians may find other and faster ways to produce the adaptive parametrization, and they may also improve the computation of the displacements of poles to accelerate the convergence or produce a real minimax approximation.

References

1. Bézier, P., Essai de définition numérique de courbes expérimentales, Thèse de doctorat d'état, Univ. Pierre et Marie Curie, Paris, 02/1977.

2. Brun, J. M., S. Foufou, A. Bouras, and P. Arthaud, In search of an optimal parametrization of curves, in *Proceedings of the fourth Int'l Conf. On computational graphics and visualisation techniques*, Lisbon, Portugal, Dec 11-15, 1995.
3. Bogacki, P., E. W. Stanley, and X. Yuesheng, Degree reduction of Bézier curves by uniform approximation with endpoint interpolation, Computer-Aided Design **27,9** (1995), 651–661.
4. Brunnett, G., T. Schreiber, and J. Braun, The geometry of optimal degree reduction of Bézier curves, Comput. Aided Geom. Design **13** (1996), 773–788.
5. Eck, M., Degree reduction of Bézier curves, Comput. Aided Geom. Design **10** (1993), 237–251.
6. Eck, M., Least square degree reduction of Bézier curves, Computer-Aided Design **27,11** (1995), 845–851.
7. Farin, G., *Curves and Surfaces for CAGD a Practical Guide*, Academic Press, 1997.
8. Foufou, S., Brun, J.M., and A. Bouras, Study of iterative processes convergency, in *Proceedings of the third int'l conf. in central europe on computer graphics and visualization*, Plzen, Czech Rep., Feb 14-18, 1995.
9. Harada, K., and E. Nakamae, Application of the Bézier curve to data interpolation, Computer-Aided Design **14,1** (1982), 55–59.
10. Hoscheck, J, Intrinsic parametrization for approximation, Comput. Aided Geom. Design **5** (1988), 27–31.
11. Lachance, M., Chebyshev economization for parametric surfaces, Comput. Aided Geom. Design **5** (1988), 195–208.
12. Lee, E., Choosing nodes in parametric curve interpolation, Computer-Aided Design **21,6** (1984), 363–370.
13. Wahu, G., Critical criteria for data reduction in curve and surface models conversion, in *Proceedings of the first Swiss Conference on CAD/CAM*, Neuchâtel, Switzerland, Feb 22-24, 1999.
14. Watkins, M. A., and A. J. Worsey, Degree reduction of Bézier curves, Computer-Aided Design **20,7** (1988), 398–405.

G. Wahu and A. Bouras
LIGIM, University Claude Bernard Lyon 1,
43 boulevard du 11 novembre 1918
69622 Villeurbanne, France
gwahu@ligim.univ-lyon1.fr

J.M. Brun
LIM/XAOLab, University Aix-Marseille 2
ESIL, Case 925
13288 Marseille cedex 9, France
jmbrun@esil.univ-mrs.fr

On the Geometry of Sculptured Surface Machining

Johannes Wallner and Helmut Pottmann

Abstract. We present geometric aspects of sculptured surface machining. Several possible configuration manifolds of tool positions relative to a workpiece are investigated under different aspects: the degree of freedom of the motion of the tool, the correspondence between the contact point and the tool position, and the presence or absence of unwanted collisions between tool and workpiece.

§1. Introduction

In the past decades, strong research efforts have been devoted to developing the mathematical fundamentals and efficient algorithms for the representation of free-form surfaces in CAD/CAM systems. However, just a few contributions address *manufacturing* of sculptured surfaces, although there are appealing and practically important open problems in this area.

Geometrical problems in this area include the following: If two surfaces touch each other at a point, such as a milling-tool and a free-form surface which is to be manufactured, does the curvature of the surfaces force them to intersect arbitrarily near the contact point (= the local collision problem)? Given a free-form surface and a milling tool, is the tool able to move such that its envelope during the motion is the free-form surface? Is the tool able to do this while moving only by translations (3-axis milling), or do we need more flexibility (5-axis milling)? Which relative tool position achieves best surface quality (=tool positioning)? How can we decompose the theoretically two-parameter motion of the milling tool by a series of one-parameter motions such that e.g. manufacturing time is minimal (=motion planning)?

A survey of mathematical fundamentals on NC machining of sculptured surfaces is given in [1,14,19]. Tool selection, motion planning and local interference checking for 3-axis and 5-axis machining has been studied in [2,9,10,11, 12,13,25]. Three-axis machining (general offsets and Minkowski addition) are considered by [15,21,23]. A configuration space has been defined in [3,24].

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 417–432.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

There is a variety of contributions using Computer Graphics techniques such as visibility algorithms [5,7,22]. There the cutter shaft is shrunk to its axis, and simultaneously the design surface is replaced by an appropriate offset surface. Collision checking is thus transformed to a visibility test (5-axis machining requires a modification of the method).

This paper is organized as follows: In Section 2 we briefly show some concepts of elementary differential geometry. Section 3 sums up some results concerning the local contact situation. Section 4 investigates configuration manifolds of motions constrained in several ways and describes the possible infinitesimal motions of a milling-tool in a contact position. In Section 5 we study the dependency of the contact *point* of the contact *position*. Finally Section 6 features global statements about the absence of unwanted collisions under certain circumstances.

§2. Differential Geometry

We first give a short description of some aspects of curvature theory of 2-surfaces in Euclidean three-space (cf. [4,17,20]).

2.1. Oriented surfaces and their first and second fundamental forms

Consider a regular smooth surface given by the parametrization $f = f(u)$, where $x = (x_1, x_2, x_3)$ is a point in Euclidean $\mathbb{R}^3$ and $u = (u_1, u_2)$ ranges in some open planar domain D. We assume that f is twice continuously differentiable in order to be able to define curvatures.

The differential df of f maps a tangent vector $v = (v_1, v_2)$ attached to the point u, to the vector $d_u f(v) = \frac{d}{dt}|_{t=0} f(u+tv)$, which is computed by $d_u f(v) = x_{,1}(u)v_1 + x_{,2}(u)v_2$, where the symbols $x_{,1}$ and $x_{,2}$ mean differentiation with respect to the first and second variable.

The function $n = (x_{,1} \times x_{,2})/\|x_{,1} \times x_{,2}\|$ is the surface unit normal vector. The symmetric bilinear forms $g_u(v,w) = d_u f(v) \cdot d_u f(w)$ and $h_u(v,w) = -dn(v) \cdot w$ are called the first and second fundamental forms of f. With $g_{ij} = x_{,i} \cdot x_{,j}$ and $h_{ij} = -n_{,i} \cdot x_{,j} = n \cdot x_{,ij}$ we have $g(v,w) = \sum g_{ij} v_i w_j$ and $h(v,w) = \sum h_{ij} v_i w_j$.

2.2. The Dupin indicatrix and Meusnier's theorem

If f is the parametrization of a surface, and $u = u(t)$ is a curve in its parameter domain D, then $c(t) = f(u(t))$ is a curve contained in the surface $f(D)$. The curve's tangent $\dot{c}(t) = df(\dot{u}(t))$ is contained in the surface's tangent plane. Its second derivative vector is split into three components:

$$\ddot{c} = \|\dot{c}\| \left(\kappa_n n + \alpha \dot{c} + \kappa_g b\right),$$

where n is the normal vector evaluated at $u(t)$ and b is the curve's normal in the tangent plane. The coefficients κ_g and κ_n are the geodesic curvature and normal curvature of the curve, respectively. The following theorem states the perhaps unexpected fact that the normal curvature is dependent only on the direction $\dot{c}$, and we can therefore speak of the normal curvature of a surface tangent.

Theorem 1. *(Meusnier) The normal curvature of the curve c is computed by* $\kappa_n = \kappa_n(\dot{u}) = h(\dot{u},\dot{u})/g(\dot{u},\dot{u})$.

Evaluate the matrix product $(g_{ij})^{-1}h_{ij}$ at (u_1,u_2), and compute its two linearly independent eigenvectors v', v'' which correspond to eigenvalues κ_1, κ_2. Then $df(v')$, $df(v'')$ are orthogonal and define the two principal surface tangents at u.

Theorem 2. *(Euler) Assume that w', w'' are unit vectors parallel to the principal surface tangents at u. If $df(v) = \cos\phi w' + \sin\phi w''$, then $\kappa_n(v) = \cos^2\phi \cdot \kappa_1 + \sin^2\phi \cdot \kappa_2$. The polar diagrams of $1/\sqrt{\kappa_n}$ and $1/\sqrt{-\kappa_n}$ in the tangent plane (the oriented* Dupin indicatrices *i_+, i_-) are possibly void or singular conic sections centered in the origin.*

Surface points are called elliptic, if $\kappa_1\kappa_2 > 0$, hyperbolic if $\kappa_1\kappa_2 < 0$, flat if $\kappa_1 = \kappa_2 = 0$, and parabolic in the remaining cases.

2.3. Euclidean displacements and infinitesimal motions

A Euclidean displacement $g : x \in \mathbb{R}^3 \mapsto g(x) \in \mathbb{R}^3$ may be written in the form $x \mapsto M \cdot x + v$, where M is an orthogonal matrix of determinant 1. A one-parameter family $g(t) = (v(t), M(t))$ of Euclidean displacements, (= a path of Euclidean motions, or a smooth Euclidean motion) has in all of its instants an infinitesimal motion, which is determined by the velocity vectors $d(g(t))(x)/dt = \dot{v}(t) + \dot{M}(t)\cdot x$ of all points. If an infinitesimal motion coincides with the velocity field of a smooth rotation about an axis, it is called an infinitesimal rotation. The definition of infinitesimal translation and infinitesimal helical motion is analogous.

There is a linear space of infinitesimal motions. It is further well known that all infinitesimal motions (i.e, all velocity fields of smooth motions) can be written in the form

$$\dot{x} = \bar{c} + c \times x.$$

The condition $c\cdot\bar{c} = 0$ characterizes infinitesimal rotations, $c = 0$ characterizes infinitesimal translations, and $c \neq 0$, $c\cdot\bar{c} \neq 0$ characterizes infinitesimal helical motions. We briefly write $(c,\bar{c}) \in \mathbb{R}^6$ to denote an infinitesimal motion. An infinitesimal rotation whose axis is $a + [b]$ then has the form $\lambda(b, a \times b)$ (the symbol $[b]$ denotes all multiples of the vector b). The translation $\dot{x} = \bar{c}$ has the form $(0,\bar{c})$. If we have to consider the coordinates of $c, \bar{c}$ in some coordinate system, we always write $c = (c_{01}, c_{02}, c_{03})$ and $\bar{c} = (c_{23}, c_{31}, c_{12})$.

2.4. Ruled surfaces

If $p(t)$, $v(t)$ are two curves with $v \neq 0$, then $f(u_1,u_2) = p(u_1) + u_2 v(u_1)$ is a surface whose parameter lines $u_1 = \text{const}$ are straight lines, which will be denoted by $l(u_1)$. Such a surface is called a ruled surface.

We need the following well known results concerning the first order differential properties of ruled surfaces in Euclidean space: there is an orthonormal frame $(q; e_1, e_2, e_3)$, dependent on a parameter t, and a smooth function $u_1(t)$

such that $q(t)$ is on the ruling $l(u_1(t))$, $e_1(t)$ is parallel to it, and $e_2(t)$ is tangent to the surface at $q(t)$, and which can be chosen such that its infinitesimal motion at $t = 0$ is a helical motion $(c, \bar{c})$ which can be computed by the following formulae:

$$\bar{p} := p \times v \quad s := \frac{1}{\|p \times \dot{p}\|^2} (\det(p, \dot{p}, \dot{\bar{p}})p + \det(p, \bar{p}, \dot{p})\dot{p} + (\bar{p} \cdot \dot{p})(p \times \dot{p}))$$

$$b = \frac{(\dot{p} \times v) \times v}{\|(\dot{p} \times v) \times v\|}\Big|_{t=0}, \quad \delta = \frac{(\dot{p}\dot{\bar{p}})p^2}{(p \times \dot{p})^2}\Big|_{t=0}, \quad c = \delta b, \quad \bar{c} = b - c \times s(0).$$

The first order differential invariant δ is called the distribution parameter of the ruled surface, and the point $s(t)$ is called its striction point.

2.5. Line congruences

A smooth line congruence $\mathcal{K}$ is a smooth two-parameter family of straight lines $l(u_1, u_2)$ in Euclidean three-space. It may be parametrized by two 'surfaces' $p(u_1, u_2)$ and $v(u_1, u_2)$, where $v(u_1, u_2) \neq 0$. The line $l(u_1, u_2)$ then is $p(u_1, u_2) + [v(u_1, u_2)]$. The choice of a curve $(u_1(t), u_2(t))$ in the parameter domain gives a ruled surface $l(u_1(t), u_2(t))$.

Definition. *A line congruence $\mathcal{K}$ is* regular *at a line l if the six-tuples $(p, p \times v)$, $(p_{,1}, p \times v_{,1} + p_{,1} \times v)$, $(p_{,2}, p \times v_{,2} + p_{,2} \times v)$ are linearly independent.*

The meaning of this definition is that the lines of the congruence actually change infinitesimally if we move infinitesimally in the parameter domain. We will always assume that $\mathcal{K}$ is regular.

It is well known that the infinitesimal properties of first order of a line l within $\mathcal{K}$ are like those of a *linear* line congruence $\mathcal{K}'$, which is called the *tangent* to $\mathcal{K}$. There are the following possibilities for $\mathcal{K}'$:

- $\mathcal{K}'$ is the set of lines which intersect two lines k', k'' (= a hyperbolic linear congruence with axes k', k'').
- $\mathcal{K}'$ is the set of lines whose complex extensions intersect two conjugate complex lines k', k'' (= an elliptic linear congruence). This set of lines is the affine image of the set of lines which join the points $(x, y, 0)$ and $(x \cos\phi - y \sin\phi, x \sin\phi + y \cos\phi, 1)$, $0 \leq \phi < 2\pi$.
- $\mathcal{K}'$ is the set of lines tangent to a ruled quadric in the points of one of its rulings k. (= a parabolic linear congruence). The line k is also called the axis of $\mathcal{K}'$.
- $\mathcal{K}'$ is a bundle of lines.

In all cases, lines at infinity are allowed (but in the elliptic case they do not occur). These four types of lines l in a congruence are accordingly called hyperbolic, elliptic, parabolic, and degenerate.

In the hyperbolic and parabolic case, the points of l contained in an axis are called focal points, in the degerenate case the bundle vertex is also called a focal point.

There is also a connection between infinitesimal motions and linear line congruences: a four-dimensional subspace of the space of infinitesimal motions always contains infinitesimal rotations or translations, whose axes form a linear line congruence (here we assign to the infinitesimal translation $\dot{x} = \bar{c}$ the axis at infinity which is orthogonal to $\bar{c}$).

2.5 Submanifolds

As the concepts listed here are essential only for the proofs of our results, we give only a brief summary: We assume that the reader is familiar with the concept of a smooth n-dimensional manifold N (n-manifold). An embedded smooth m-submanifold M of N is characterized by the existence, for $p \in M$, of a local diffeomorphism which transforms an N-neighbourhood U of p to $\mathbb{R}^n$ and $U \cap M$ to $\mathbb{R}^m \subset \mathbb{R}^n$. An immersion is a smooth mapping whose differential is one-to-one (but not necessarily onto). Then locally the immersion is also one-to-one. An immersed k-submanifold is the image of a smooth k-manifold under an immersion. The difference between embedded and immersed submanifolds is therefore that the latter may have 'self-intersections', but neither are allowed to have 'singularities'.

An embedded submanifold M_1 and an immersed submanifold M_2 of N are transverse (we write $M_1 \pitchfork M_2$), if for all points $p \in M_1 \cap M_2$ the tangent spaces T_pM_1, T_pM_2 span T_pN. Then $M_1 \cap M_2$ is an immersed $(\dim M_1 + \dim M_2 - \dim N)$-dimensional submanifold of N, whose tangent space equals $T_pM_1 \cap T_pM_2$.

If a smooth mapping ϕ of a smooth m-manifold M into a smooth n-manifold N has constant rank r (i.e., at all points its differential's rank as of a linear mapping equals r), then $\phi(M)$ is a smooth immersed r-dimensional submanifold of N.

§3. Local Contact Situation

If a body is bounded by a smooth surface f, this surface has an inside and an outside. The unit surface normals can point to either side, depending on the parametrization. If two bodies, which are bounded by smooth surfaces f', f'', touch each other, the curvatures of f', f'' give information whether they intersect locally or not.

It should be remarked that some methods proposed in the literature (cf. [10,12]) for avoiding local intersections are only approximations, and one can find surfaces where they won't work. Also it is important to note that the presence or absence of local intersections is completely independent of the actual motion of the two bodies.

Definition. *The interior* $\mathrm{int}(i)$ *of a conic* i *centered in the origin is void if* i *is void, and otherwise is the connected component of* $\mathbb{R}^2 \setminus i$ *which contains* 0 *and whose boundary is* i. *Its exterior* $\mathrm{ext}(i)$ *is the complement of* $\mathrm{int}(i) \cup i$.

We assume that f', f'' are parametrized such that their unit normal vectors in the common point coincide, and that this common normal vector points to

the inside for f' and to the outside for f''. We further assume that not all four principal curvatures are zero. We consider the Dupin indicatrices i'_+, i'_-, i''_+, and i''_- for f' and f'', respectively.

Theorem 3. *Under these assumptions, the two bodies in question intersect locally, if one of the intersections $i''_+ \cap \mathrm{int}(i'_+)$, $i''_- \cap \mathrm{ext}(i'_-)$ is not void. They do not intersect, if $i''_+ \cap i'_+$, $i''_+ \cap \mathrm{int}(i'_+)$, $i''_- \cap i'_-$, $i''_- \cap \mathrm{ext}(i'_-)$ are void.*

Proof: (Sketch) The theorem follows from the fact that a twice continuously differentiable surface may be approximated of second order by the graph of a quadratic function, whose contour lines are scaled versions of the Dupin indicatrices. □

Note that the theorem says nothing about the cases that all principal curvatures are zero, the indicatrices touch each other in two opposite points, or even coincide (cf. Fig. 1, right). In that case, second derivatives are not sufficient to decide if there are local intersections. In practice, this does not matter very much because the only case that is likely to occur with nonzero probability is that of a flat end mill shaping a planar surface, which does not have self-intersections.

In [8,16,23] it is shown how to define indicatrices in the case of *piecewise* curvature-continuous surfaces. The theorem is valid also in this more general case.

§4. Configuration Manifolds and their Tangent Spaces

For many problems concerning the milling of free-form surfaces, it is important to know the degree of freedom of a motion constrained in various ways. Typical constraints are: Motion by translations such that a surface remains in contact with another surface (3-axis milling), motion such that a surface remains in contact with another surface (possible set of tool positions in 5-axis milling), motion such that a milling tool remains in contact with a surface and its axis is contained in some previously prescribed line congruence (a possible way to do 5-axis milling). These topics will be discussed in Subsections 4.1–4.3.

4.1 Translational motions constrained by surface-surface contact

Consider two surfaces f', f'' which have a common point $p = f'(u') = f''(u'')$ and share a common unit surface normal $n' = n''$ there. Imagine the first surface moving by translations such that it always touches the second surface (i.e., at every instant t there are a translation vector $a(t)$ and parameter values $u'(t)$ and $u''(t)$ such that $f'(u'(t)) + a(t) = f''(u''(t))$). One would expect that this motion has two degrees of freedom, if we do not count intersections of the surfaces.

If $q(x_1, x_2) = \sum a_{ij} x_i x_j$ is a bivariate homogeneous quadratic polynomial in the variables x_1, x_2, we call the rank of the (2×2)-matrix a_{ij} the rank of q. The zero set of q consists of the entire plane in the case of zero rank, of one line if the rank is one, and of two real or two conjugate complex lines if the rank is two.

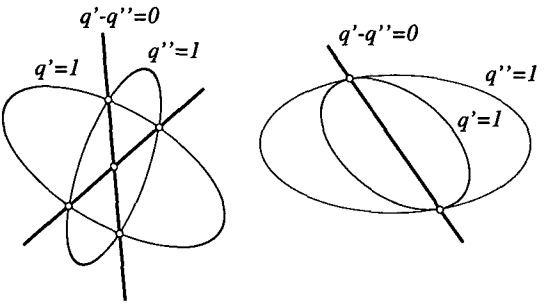


Fig. 1. Indicatrices $q' = 1$, $q'' = 1$ and zero set of $q' - q''$. Left: $\mathrm{rk}(q' - q'') = 2$, Right: $\mathrm{rk}(q' - q') = 1$. .

We have the following theorem (cf. [8,16,23]), which is valid for all contact situations where at least one surface has no parabolic or flat point, or both have parabolic points but the principal tangents do not coincide.

Theorem 4. *Write the equation of the oriented Dupin indicatrices of f', f'' at the contact point in the form $q'(x_1, x_2) = 1$ and $q''(x_1, x_2) = 1$, where x_1, x_2 are Cartesian coordinates in the tangent plane, and q', q'' are bivariate homogeneous quadratic polynomials in x_1, x_2. If the condition stated immediately before this theorem is satisfied, then the rank of $q' - q''$ gives the infinitesimal degree of freedom of translational motions constrained by the contact of f', f'' (see Fig. 1).*

Proposition. *If, under the assumptions of Th. 4, the infinitesimal degree of freedom is two, then so is the local degree of freedom.*

4.2 Motions constrained by surface-surface contact

Definition. *The set of proper Euclidean motions which transforms a surface f' such that it touches a surface f'' is called the* configuration space $C = C(f', f'')$ *of surface-surface contact.*

Clearly, a position $g \in C$ is not determined by the contact points alone, because we still may rotate f' about the contact normal. But if we prescribe a unit tangent vector $(p'; w')$ of f' and $(p'; w'')$ of f'' (which means $p' = f'(u_1', u_2')$ and $w' = df'(v')$ with $\|w'\| = 1$, and the same for (p'', w'')), then there is a unique Euclidean motion $g \in C$ which maps not only the point p' onto p'', but also the tangent vector w' onto w''. If we rotate both w', w'' about an angle ϕ, this leads to the same $g \in C$, so C is a smooth image of the factor manifold $\tilde{C}$ of such equivalence classes of unit tangent vectors, which is five-dimensional.

This shows that we may expect five degrees of freedom, if f' moves under the constraint that it touches f'' in some point. The following theorem is given in [18,24]:

Theorem 5. *In the notation of Th. 4, the motion of a surface under the single constraint that it touches a second one, has five degrees of freedom if $q' - q''$ has rank two. If that is the case for all possible contact points, then the configuration space is an immersed five-dimensional submanifold of the motion group.*

Note that the theorem is valid without the additional assumption made in Th. 4 in case some principal curvatures are zero. It is easy to describe the tangent space T_gC of the configuration manifold at a contact position $g \in C \subset G$ (to be precise, the tangent space of the immersion described at the beginning of Sect. 4.2). The proof of the following proposition can be found in [18].

Proposition. *The linear space T_gC of infinitesimal motions which belong to paths in C is five-dimensional, and the axes of its infinitesimal rotations are the lines which intersect the contact normal.*

In a Cartesian coordinate system whose origin is the contact point and whose x_3-axis is the contact normal, T_gC has the equation $c_{03} = 0$.

4.3. Motions constrained by congruences

Here we consider the motion of a rigid body Σ such that a line a of Σ is contained in a congruence $\mathcal{K}$. This subset of the group G of proper Euclidean motions will be denoted by K. We assume that $\mathcal{K}$ is parametrized by $l(u_1, u_2) = p(u_1, u_2) + [v(u_1, u_2)]$.

If $g \in K$, then it is obvious that both $\tau \circ g$ and $\rho \circ g$ are in K, if τ is a translation parallel to $g(a)$ and ρ is a rotation with axis $g(a)$.

Lemma. *Assume that $\mathcal{K}$ is a smooth line congruence which is regular at $g(a)$. Then K is a four-dimensional smooth submanifold of G in a neighbourhood of g.*

Proof: (Sketch) Let $g \in K$ be a position of Σ such that $g(a) \in \mathcal{K}$. We compute K's tangent space T_gK of infinitesimal motions at g: Consider a curve $(u_1(t), u_2(t))$ in $\mathcal{K}$'s parameter domain such that $l(t) = l(u_1(t), u_2(t))$ is a ruled surface within $\mathcal{K}$ with $l(0) = g(a)$. The helical motion described in Sect. 2.4 is tangent to K. If we choose two such curves with linearly independent tangent vectors, this gives two linearly independent infinitesimal motions of T_gK, if $\mathcal{K}$ is regular at $g(a)$ (the proof of this is left as an exercise to the reader).

Obviously all infinitesimal translations parallel to $g(a)$ are in T_gK, and so are the infinitesimal rotations with axis $g(a)$. If $g(a)$ is the line $p + [v]$, then the former is described by the six-tuple $(0, v)$ and the latter by $(v, p \times v)$. Now T_gK is the linear span of these four infinitesimal motions. □

In case that $g(a)$ is a hyperbolic line of $\mathcal{K}$, there is a simple geometric characterization of the infinitesimal rotations of T_gK:

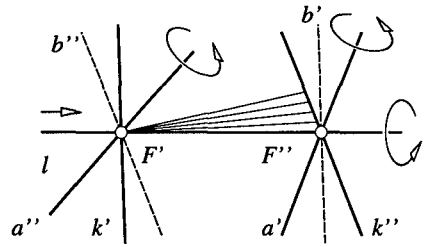


Fig. 2. Axes of infinitesimal rotations in K.

Proposition. *If $\mathcal{K}'$ is a hyperbolic linear congruence tangent to $\mathcal{K}$ at the line $g(a)$ and has axes k', k'', then the axes of infinitesimal rotations in T_gK form a hyperbolic linear congruence $\mathcal{L}$, whose axes b', b'' are incident with $g(a) \cap k''$, $g(a) \cap k'$, respecively, are orthogonal to $g(a)$, and are such that $k', b', g(a)$ as well as $k'', b'', g(a)$ are coplanar (see Fig. 2).*

Proof: It is easy to see that the lines a', a'', incident with $k' \cap g(a)$, $k'' \cap g(a)$ and orthogonal to k', k'', respectively, are axes of infinitesimal rotations contained in T_gK. The axis of the infinitesimal translations along $g(a)$ (which is the line at infinity orthogonal to $g(a)$), $g(a)$ itself, and a', a'' intersect both b', b''. We already know that the set of axes is a linear congruence; because of these four intersections the lines b', b'' are necessarily the axes of $\mathcal{L}$. □

The following is used later:

Proposition. *A point of $g(a)$ which contains two different axes of infinitesimal rotations of T_gK must be a focal point of $g(a)$.*

Proof: At least one of the two axes is not $g(a)$ itself, and the rotation about this axis transforms $g(a)$ into a line which intersects $g(a)$. Looking at the list of linear tangent congruences in Sect. 2.4 shows that this is only possible in a focal point. □

4.4. Multiple constraints

Assume that a rigid body Σ, bounded by a smooth surface f', moves such that f' remains in contact with a surface f'', and that in addition a line a of Σ is contained in a smooth line congruence $\mathcal{K}$.

With C being the configuration space of surface-surface contact (see Sect. 4.2) and K as in Sect. 4.3 the set of possible positions g of Σ is given by the intersection $C \cap K$. The following theorem shows under what circumstances $C \cap K$ is actually a smooth three-dimensional submanifold of positions, as is to be expected when comparing dimensions:

Theorem 6. *If T_gK is not contained in T_gC, then $C \cap K$ is a three-dimensional immersed submanifold in a neighbourhood of g. This is always the case if $g(a)$ is not parallel to the contact tangent plane.*

Proof: We have to show that $C \pitchfork K$. Because $\dim T_gC = 5$, this is always the case if T_gK is not contained in T_gC. If the contact tangent plane is not

parallel to l, then any small translation parallel to l leaves C and therefore in this case $C \pitchfork K$, and $\dim(C \cap K) = \dim C + \dim K - \dim G = 3$. □

If Σ has rotational symmetry, and the line $g(a)$ which is forced to belong to $\mathcal{K}$ is its axis, then we can say more about the set $C \cap K$ of admissible positions. Clearly, any rotation about the axis does not change Σ, so $g \in C \cap K$ implies that $\rho \circ g \in C \cap K$ whenever ρ is such a rotation.

We choose a reference point p on the axis, and look at the three-parameter family of its positions, which is actually only a two-parameter family:

Proposition. *If Σ has rotational symmetry, a reference point p in Σ's axis a traces out a regular two-surface while undergoing all transformations of $C \cap K$, provided that p is never a focal point of $g(a)$. This path surface is transverse to l if l is not parallel to the contact tangent plane.*

Proof: We consider the mapping $\phi : g \mapsto g(p)$ of $C \cap K$ to $\mathbb{R}^3$. All infinitesimal rotations about $g(a)$ assign zero velocity to p, which implies that the rank of ϕ is not greater than two. If p has zero velocity also for other infinitesimal motions of $T_g(C \cap K)$, these must be infinitesimal rotations, and we can use the proposition at the end of Sect. 4.3 to conclude that the rank of ϕ is indeed two, and its image a smooth 2-surface. If l and the contact tangent plane are not parallel, then no infinitesimal translation parallel to l is in $C \cap K$, and the path surface cannot be parallel to l. □

§5. Movement of the Contact Point

It is important to study the dependency of the contact point on the contact positions. This will be done for two different types of constraints.

5.1. Motions constrained by surface contact

In Sect. 4.2 we stated that the configuration space $C(f', f'')$ of surface-surface contact is an immersed image of $\widetilde{C}$ as described in Sect. 4.2, if the Dupin indicatrices of f', f'' fulfill a certain condition (Th. 5).

If this is the case, then there is a local inverse $C \to M$, and so the contact point depends on the positions $g \in C$ in a smooth way locally. This (local) mapping will be denoted by ψ.

As $\dim C = 5$ and the contact point varies in a 2-surface, there is a 3-dimensional kernel subspace $\ker d_g\psi \subset T_gC$ of infinitesimal motions which do not (infinitesimally) change the contact point. The following is an easy exercise in differential geometry:

Lemma. *If f', f'' are two surfaces having contact at a point p, n is the unit normal vector of f', and w_1, w_2 are principal tangent vectors, corresponding to curvatures κ_1, κ_2, then all infinitesimal rotations about the axes $p + [n]$, $p + n/\kappa_1 + [w_2]$, $p + n/\kappa_2 + [w_1]$ are contained in the tangent space T_gC of the configuration manifold, and do not (infinitesimally) change the contact point on f''.*

If a principal curvature is zero, the corresponding axis will be at infinity. Obviously rotations of these three types span $\ker d_g\psi$.

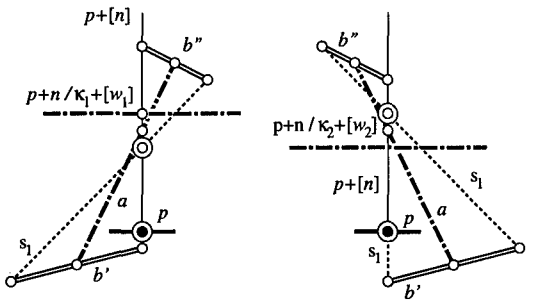


Fig. 3. Situation where the contact point has only one infinitesimal degree of freedom (cf. Section 5.2): front view and lateral view.

5.2. Multiply constrained motion

Now we consider the configuration manifold $C \cap K$ of a motion constrained by surface-surface contact and a line congruence $\mathcal{K}$ as described in Sect. 4.4. The contact point still depends smoothly on $g \in C \cap K$. But does it have nonzero velocity for all infinitesimal motions different from the rotations about the contact normal? Obviously, that depends on the intersection $\ker d_g\psi \cap T_g(C \cap K)$. Because $\ker d_g\psi \subset T_gC$, this intersection equals $\ker d_g\psi \cap T_gK$. Depending on its dimension, there are the following three possibilities:

- dim = 1: Only the infinitesimal rotations about the contact normal are in $\ker d_g\psi$. The infinitesimal motion of the contact point is two-dimensional.
- dim = 2: The rank of ψ is one, and the contact point varies infinitesimally only in one direction.
- dim = 3: $\mathrm{rk}\psi = 0$ and the contact point does not move infinitesimally.

If the line $g(a)$ belonging to the current position $g \in C \cap K$ is a hyperbolic line of $\mathcal{K}$, then the rank of ψ's restriction to $C \cap K$ can be determined geometrically (see Fig. 3):

Proposition. *We use the notation of the propositions in Sect. 4.3 and of the lemma in Sect. 5.1. If there are lines s_1, s_2 such that s_i intersects both b', b'', s_i is incident with $p+n/\kappa_i$ $(i = 1, 2)$, and $s_1 \subset p+[n]+[w_2]$, $s_2 \subset p+[n]+[w_1]$, then $\mathrm{rk}\psi|C \cap K \leq 1$, otherwise $\mathrm{rk}\psi|C \cap K = 2$.*

Proof: The lemma in Sect. 5.1 describes $\ker_g \psi =: A$ and the first proposition in Sect. 4.2 does the same for $T_gK =: B$. To compute $A \cap B$ we apply the duality which assigns to a linear space $A \subset \mathbb{R}^6$ the linear space $A^\star$ of those infinitesimal motions $(a^\star, \bar{a}^\star)$ which fulfill $a \cdot \bar{a}^\star + \bar{a} \cdot a^\star = 0$ for all $(a, \bar{a}) \in A$. Clearly, $\dim A^\star = 6 - \dim A$ and $(A \cap B)^\star$ is the linear span of $A^\star$, $B^\star$.

If is well known that an infinitesimal rotation is in $A^\star$ if and only if its axis intersects all the axes of the infinitesimal rotations of A. Thus, the rotation axes in $A^\star$ consist of two pencils with the same vertices as those of A, but with orthogonal planes; and the rotation axes of $B^\star$ are just the lines b', b''.

Rank ≤ 1 means $\dim(A^* + B^*) \leq 4$, or that the span of b' and A^* contains b''. It is easily seen that apart from even more special cases, this happens if the axes of this span are a hyperbolic linear congruence with axes s_1, s_2. □

§6. Collision Checking

It is possible that a milling tool has no local intersections with the finished surface, but while in contact with the surface at one point, it cuts into another part. Algorithms which test for this type of intersection of two bodies in space are time consuming, and therefore we want to circumvent the general collision test in some way. In some cases we are able to predict the total absence of collisions based only on the curvatures of the boundaries of the two bodies involved.

We say that a surface Φ is millable by a body Σ, if (a) there are no local intersections, and (b) it is possible for Σ to move, within previously imposed constraints, along the surface Φ such that it touches Φ during this motion, but never actually intersects it.

6.1. 3-axis milling

3-axis milling means that a milling tool Σ rotates about its axis a and moves such that a remains parallel to a fixed line, and Σ always touches the finished surface. As the rotation about a is not important for geometric considerations, we disregard it completely and consider a body which moves in a translational manner.

We assume that Σ as well as the workpiece Φ are bounded by piecewise twice continuously differentiable surfaces (convex edges are allowed). Then the so-called general offset surface (defined below, see Fig. 4, right) of Φ with respect to Σ shows in its singularities and self-intersections the singular positions of the motion and the collisions (cf. [15,16]).

Definition. *Choose a reference point p attached to Σ. Consider the set of translations τ such that $\tau(\Sigma)$ touches Φ (disregarding intersections). Then the set of all points $\tau(p)$ is called the* general offset surface *of Φ with respect to Σ.*

Theorem 7. *If Σ is strictly convex with positive principal curvatures, and Φ is connected, then the general offset surface of Φ with respect to Σ is smooth. It is regular in all points which correspond to translationally regular contact positions, and if it is both regular and free of self-intersections, then Φ is globally millable by Σ.*

Proof: (cf. [16,23]) For all points $p \in \Phi$ there is a unique position g such that $g(\Sigma)$ touches Φ in p. The parametrization of the general offset which thus is induced by the parametrization of Φ is easily seen to be smooth. There is always a point p_0 such that $\Sigma \cap \Phi$ consists of p_0 only. If Σ touches Φ in p, and thereby cuts into another part of Φ, then let Σ move such that the contact point follows a curve which joins p_0 with p. At the first time that the set

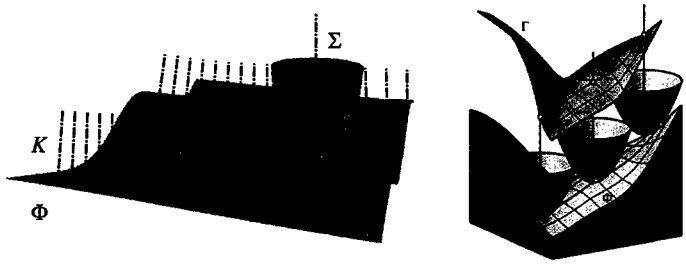


Fig. 4. Left: Line congruence $\mathcal{K}$, finished workpiece Φ and cutter Σ. Right: Workpiece Φ, milling-tool Σ and general offset surface Γ.

$\Sigma \cap \Phi$ consists of more than one point, Σ and Φ touch each other in all points of $\Sigma \cap \Phi$, which leads to a multiple point of the general offset. □

Thus, we have transformed the collision problem into the problem of determining singularities and self-intersections of certain surfaces. If Φ has some additional properties then it is not difficult to guarantee total absence of unwanted collisions provided that no local collisions occur:

- if Φ is convex and Σ, Φ are oppositely oriented (i.e., the body bounded by Φ is its inside); or
- if Φ is convex and Σ, Φ are equally oriented (i.e., the body bounded by Φ is its outside and Σ is inside); or
- if Φ is star-shaped and bounds its inside; or
- if Φ is star-shaped, bounds its outside, and Σ fits into the convex core of Φ (the convex core of a star-shaped set M is the set of points with respect to which M is star-shaped — it is a convex subset of M); or
- if Φ is the graph surface of a function over a planar domain whose boundary is millable by the 'top view' contour of Σ.

Proof: The proof of this can be found in [16], and generalizations actually unimportant for applications are studied in [23]. The idea of the proof, which is important also for the proof of Th. 8, is as follows: We assume that there is a 'projection' of entire space onto Φ. If e.g. Φ is convex, just choose any interior point o, and to project a point p, intersect the ray $\overrightarrow{op}$ with Φ.

Then consider the following mapping whose domain is Φ: For a point $x \in \Phi$, translate Σ such that it touches Φ there, and project a previously chosen reference point of Σ onto Φ. This mapping is shown to be smooth and orientation-preserving, and by scaling Σ with a factor λ $(1 \geq \lambda \geq 0)$ is deformed into the identity mapping. Differential topology allows now to conclude the Φ is one-to-one and onto, which means that no translate of Σ can touch Φ in more than one point. □

6.2. 5-axis milling constrained by a line congruence

Here 5-axis milling means that the milling tool moves such that it touches the finished surface, and its axis is always contained in a line congruence $\mathcal{K}$.

Definition. *A regularity domain of $\mathcal{K}$ is an open domain M in $\mathbb{R}^3$ such that $\mathcal{K}$ defines a fibration of M (every point of M is contained in exactly one set $l \cap M$, where l is a line of the congruence). A section of a regularity domain M is a surface which intersects all sets $l \cap M$ exactly once. The regularity domain M is a tubular neighbourhood of its section Φ if it is diffeomorphic to $\Phi \times \mathbb{R}$.*

The domain M is a tubular neighbourhood of Φ if the sets $l \cap M$ are open line segments, the point $p = \Phi \cap l \cap M$ is an interior point of this segment, and the initial and end points of the segment depend smoothly on p.

We say that a convex body Σ with rotational symmetry is admissible for a line congruence $\mathcal{K}$ and a connected closed surface Φ which is the boundary of a subset of $\mathbb{R}^3$ if the following is fulfilled:

- There is a regularity domain M for $\mathcal{K}$ which is a tubular neighbourhood of Φ.
- Σ moves such that it is entirely contained in the regularity domain.
- In no position of $C \cap K$ the contact tangent plane is parallel to the axis of Σ.
- The contact point has two infinitesimal degrees of freedom for all scaled versions $\lambda\Sigma$, $0 \leq \lambda \leq 1$ (cf. Sect. 5.2).

These conditions are actually easy to fulfill in practice except for the last one, which is difficult to detect in advance. In the special case of three-axis milling ($\mathcal{K}$ is a bundle of parallel lines), this requires that all contact points on Σ are elliptic surface points.

Suppose that we are given a surface Φ and a milling-tool Σ, and we have chosen a congruence $\mathcal{K}$, and have found, for all contact points on Φ, a position $g(\Sigma)$ such that $g(\Sigma)$'s axis is in $\mathcal{K}$. Suppose we have already tested for local millability and the admissibility conditions described above. Then we have the following

Theorem 8. *Under the circumstances described above, the cutter does not interfere with the surface Σ, i.e., Φ is globally millable by Σ.*

Note that in many cases it will be sufficient to check the admissibility only for the cutter head, because collisions of the cutter shaft with the workpiece will be treated by different methods (see the introduction)

Proof: In Section 5 we have established that the contact point depends smoothly on Σ's position g. Let a be Σ's axis, and let ρ be a rotation about $g(a)$. Clearly $\rho \circ g$ is again a contact position with the same contact point as g. If we choose a reference point p on Σ's axis and inside Σ, the path surface of p also depends smoothly on g, and $\rho \circ g(p) = g(p)$.

This means that the contact point depends smoothly on the position $g(p)$ of the reference point. The last admissibility condition ensures that also $g(p)$ depends smoothly on the contact point. Thus, we can define a smooth mapping $f : \Phi \to \Phi$ as follows: A contact point $q \in \Phi$ is mapped to the corresponding point $g(p)$, which is subsequently mapped to the intersection

$f(q)$ of $g(a)$ with Φ. If we scale Σ with a factor λ $(0 \leq \lambda \leq 1)$, we get mappings f_λ, where f_0 is the identity mapping of Φ onto itself, and $f_\lambda(q)$ depends continously on λ. f_λ is never singular since we are in a regularity domain, $f_1 = f$ is orientation-preserving because f_0 is, and the number of pre-images of a point is the same for $f_0 = \mathrm{id}$ and $f_1 = f$. This shows that f is one-to-one and onto. If $g(\Sigma)$ touches Φ in one point and cuts into Φ in another, there also is a position $g(\Sigma)$ where Σ touches in two different points (cf. the proof of Th. 7). These two points have the same f-image by definition of f, which contradicts bijectivity of f. □

Acknowledgments. This research was supported in part by Grants P13648-MAT and P13938-MAT of the Austrian Science Fund.

References

1. Blackmore, D., M. C. Leu, L. P. Wang, and H. Jiang, Swept volume: a retrospective and prospective view, Neural, Parallel and Scientific Computations **5** (1997), 81–102.
2. Choi, B. K., J. W. Park, and C. S. Jun, Cutter location data optimization in 5-axis surface machining, Computer-Aided Design **25** (1993), 377–386.
3. Choi, B. K., D. H. Kim, and R. B. Jerard, C-space approach to tool-path generation for die and mold machining, Computer-Aided Design **29** (1997), 657–669.
4. do Carmo, M., *Differential Geometry of Curves and Surfaces,* Prentice-Hall, Englewood Cliffs, NJ, 1976.
5. Elber, G., Accessibility in 5-axis milling environment, Comput. Aided Geom. Design **26** (1994), 796–802.
6. Elber, G., Free-form surface region optimization for three- and five-axis milling, Comput. Aided Geom. Design **27** (1995), 465–470.
7. Elber, G. and E. Cohen, A unified approach to accessibility in 5-axis freeform milling environments, in *Sculptured Surface Machining 98,* G.J. Olling, B.K. Choi, R.B. Jerard (ed), Kluwer, Boston, 1998, 33–41.
8. Glaeser, G., J. Wallner, and H. Pottmann, Collision-free 3-axis milling and selection of cutting tools, Computer-Aided Design **31** (1999), 224–232.
9. Jensen, C. G. and D. C. Anderson, Accurate tool placement and orientation for finish surface machining, in *Proceedings of the symposium of Concurrent Engineering,* ASME winter annual meeting, 1992.
10. Lee, Y.-S. and T. C. Chang, Automatic cutter selection for 5-axis sculptured surface machining, Int. J. Production Research **34** (1996), 997–998.
11. Lee, Y.-S. and T. C. Chang, Admissible tool orientation control of gouging avoidance for 5-axis complex surface machining, Computer-Aided Design **29** (1997), 507–521.

12. Lee, Y.-S., Non-isoparametric tool path planning by machining strip evaluation for 5-axis sculptured surface machining, Computer-Aided Design **30** (1998), 559–570.
13. Lee, Y.-S., Mathematical modeling using different end-mills and tool placement problems for 4- and 5-axis NC complex surface machining, Int. J. Production Research **36** (1998), 785–814.
14. Marciniak, K., *Geometric Modeling for Numerically Controlled Machining*, Oxford University Press, New York, 1991.
15. Pottmann, H., General offset surfaces, Neural, Parallel and Scientific Computations **5** (1997), 55–80.
16. Pottmann, H., J. Wallner, G. Glaeser, and G. Ravani, Geometric criteria for gouge-free three-axis milling of sculptured surfaces, ASME J. of Mechanical Design **121** (1999), 241–248.
17. Pottmann, H., Wallner, J, Fundamentals of Projective and Line Geometry, lecture notes, Institut für Geometrie, Technische Universität Wien, Fall 1999.
18. Pottmann, H. and B. Ravani, Singularities of motions constrained by contacting surfaces, Mechanism and Machine Theory, to appear.
19. Radzevich, S.P., *Multi-Axis NC Machining of Sculptured Part Surfaces (Russian)*, Vishcha Shcola Publishers, Kiev, 1991.
20. Spivak, M., *A Comprehensive Introduction to Differential Geometry*, Publish or Perish, Houston, 1979.
21. Tangelder, J. W. H., J. S. M. Vergeest, and M. H. Overmars, Interference-free NC machining using spatial planning and Minkowski operations, Computer-Aided Design **30** (1998), 277–286.
22. Tseng, Y. J. and S. Joshi, Determining feasible tool-approach directions for machining Bézier curves and surfaces, Computer-Aided Design **23** (1991), 367–378.
23. Wallner, J., On smoothness and self-intersections of general offset surfaces, J. Geom., to appear.
24. Wallner, J., Configuration space for surface-surface contact, Geom. Dedicata, to appear.
25. Yang, D. C. H. and Z. Han, Interference detection and optimal tool selection in 3-axis NC machining of free-form surfaces, Computer-Aided Design **31** (1999), 303–315.

Johannes Wallner and Helmut Pottmann
Institut für Geometrie, Wiedner Hauptstraße
8–10/113, A-1040 Wien, Austria
{wallner,pottmann}@geometrie.tuwien.ac.at

Extensions: Extrapolation Methods for CAD

Hans J. Wolters

Abstract. Many operations within a solid modeling application, notably applying thickness (shelling) or blending edges (filleting), encounter difficulties during topology resolution. In order to create a solid object, certain faces have to be intersected but the geometry is such that no intersection curve can be computed. The solution is to "extend" one or both faces. This means that one has to extrapolate the underlying curves or surfaces. This operation causes instability since extrapolation is inherently an unstable process. An additional difficulty is the selection of a strategy to compute the extension amount. Furthermore, there are additional restrictions relating to continuity across the extension boundary. In this paper I will illustrate by examples some modeling situations where extensions are necessary. I present the methods currently used, and illustrate their advantages and disadvantages. Subsequently, I will demonstrate a solution for primitives such as cylinders, cones, spheres and tori. I will conclude by suggesting approaches which could avoid some of the current pitfalls.

§1. Introduction

At present almost all engineering design tasks are performed with the help of a CAD system or — more generally — mechanical design automation (MDA) software. Most all of the commercial modelling packages converged to certain standard respresentations. The geometry is represented as NURBS curves and surfaces, where truly rational representations are used only for primitives such as circular and elliptic arcs, cylinders, cones, tori and spheres. Most modelers also moved from strictly CSG representations to a hybrid model where the topology is expressed as a BRep, and the sequence of operations is stored in a CSG-like tree. As mentioned above, our focus is on solid modelling applications where it is essential to maintain a valid topological solid after each operation. We restrict ourselves here to the manifold setting.

In more technical terms we define a solid as a 3-manifold with a compact boundary which is consistently oriented. This allows us to include objects with finite surfaces without excluding objects with infinite volumes. The reader who is unfamiliar with these concepts should consult a textbook on solid modelling such as [2,3] or the excellent survey [4].

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 433–441.
Copyright © 2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

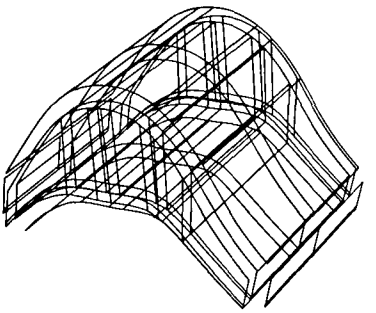


Fig. 1. Offsetting outward results in free edges which need to be resolved.

The requirement that the validity of the solid be maintained after any operation is a source for many of the robustness issues encountered in solid modelling. For example, it is often difficult to find a crisp intersection between surfaces even though the intersection curve is needed to close the solid. In this article I will focus on the problem of extensions. In a nutshell, we often need to extend surface patches in order to create intersections. This extension operator is equivalent to performing extrapolation. Very little information can be found in the literature regarding this topic; the notable exception is [5]. This survey article is meant to fill the void.

The outline of this article is as follows. In Section 2, I will describe three operations which almost always lead to the need for extending curves and surfaces. Section 3 will present the approaches used in practice and discuss their advantages and disadvantages. Furthermore, I will describe one modification which leads to significant improvements when extending quadratic Bézier patches such as cylinders, cones, spheres and tori. In Section 4 we will suggest alternative approaches to circumvent the need for extensions and encourage some future work.

§2. Extensions in Solid Modeling

The need for extensions arises quite frequently when modeling parts. I will explain the need for extensions when performing three of the most common operations, namely

- Shelling,
- Blending,
- Drafting.

These three operators are local operators, meaning that only a region of the solid is modified. Shelling is the process of applying thickness to a part. The steps to be performed are as follows: The intial step is to offset all the surfaces with prescribed offset distance, the thickness. This distance can vary considerably. The result is illustrated in Figure 1.

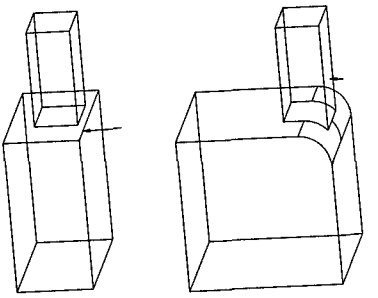


Fig. 2. Blending with features present: The edge denoted by an arrow in the left figure is filleted. The face denoted by the arrow in the right figure needs to be extended.

In Step 2, surfaces need to be intersected to form edges. Here extensions might be needed in order to compute crisp surface intersections. Trimming back the surfaces in Step 3 yields the final result. The alert reader might have noticed that the true offset is the Minkowski sum, and vertices should really correspond to arcs. This would avoid computing extensions altogether. However, this result is not desired in practice.

Blending or filleting is the process of rounding sharp edges. Hereby an additional face is constructed which meets the adjoining faces with G^1 continuity as illustrated in Figure 2. This surface is typically constructed as a loft interpolating circular or elliptic cross sections. Extensions are needed for vertex resolution when multiple filleted edges meet at a corner or for extending features when the blend face interferes with an existing feature. For more information on blending, see the survey article by Varady et al. [6].

A draft operation in solid modeling consists of changing the solid such that certain faces are slightly angled, see Figure 3. This is necessary for plastic parts manufactured by injection molding. In order to be able to pull the part out of the mold, there needs to be some room such that the faces do not stick to the mold wall. Hence this operation is mandated purely for manufacturability. Extensions are needed here as well as Figure 3 illustrates.

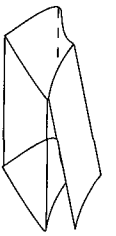


Fig. 3. Draft: The angled face replaces the front face and, hence the bottom face needs to be extended.

In conclusion, one can see that the success of these operations depends on being able to produce the geometry which is required for the successful resolution of the topology. Specifically, this means that one has to be able to produce intersection curves. The vast majority of failures can be traced to the failure of producing these curves due to bad geometry generated by extensions.

§3. Extension Methods

In this section we will survey the extension methods which are typically employed in commercial systems, and we will evaluate their strengths and weaknesses. Additionally, we will present a modification which allows to extend quadric surfaces while maintaining their current parametrization. Extensions need to fulfill certain requirements to be useful. Special surfaces such as cylinders, cones, spheres, tori should be maintained. Ideally the existing degree of continuity across the extension boundary should be kept as well; however, this requirement is mostly relaxed and only G^1 continuity is required. The shape of the resulting surface should be predictable, and the extensions should result in well-defined surface intersections. Note that the requirements differ in one crucial point: the first two requirements can be enforced, whereas the last two can not when using extrapolation based methods. We will revisit this topic in Section 4.

Subsequently, we assume as given a B-Spline surface $\boldsymbol{s}(u,v)$ of degree d with control points $\boldsymbol{s}_{ji}, i = 1,\ldots,n, j = 1,\ldots,m$ and knot vectors $\mathbf{u}$ and $\mathbf{v}$. We assume the parameter domain of the surface to be $[a_1,b_1] \times [a_2,b_2]$. We assume that we want to extend across the boundary $u = b_1$ such that the new bound is $\hat{u}$.

Natural extension

The natural extension approach is the straightforward extrapolation approach in a B-Spline or Bézier setting. Consider subdivision of a given Bézier curve $\boldsymbol{c}(t)$ with $t \in [a,b]$. It is known that the control points for any subcurve $\boldsymbol{d}(t)$ with $t \in [a,s]$ are given by the intermediate points of the de Casteljau algorithm:

$$\boldsymbol{d}_i = \boldsymbol{c}_0^i(s).$$

Of course the formula is still valid if s is lying outside the original parameter interval, here $[a,b]$. Natural extension is performed by applying this formula with a value $\hat{s} \notin [a,b]$. This results in extrapolation. Furthermore, the control points are not computed as convex combinations of the previous layer control points as before; hence attributes like the convex hull property are lost. The derivation presented here directly generalizes to the setting we are considering. In the B-Spline case the control points are the intermediate points generated by the de Boor algorithm with the new extension parameter $\hat{u}$. Pseudocode on how to compute these intermediate points can be found in the book by Farin [1]. We can easily extend this method to surfaces by repeatedly applying the

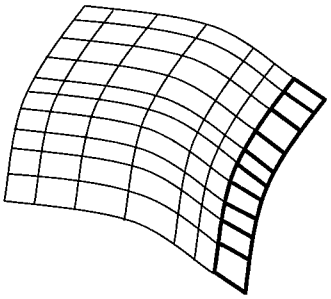


Fig. 4. Construction of linear extensions.

curve algorithm to rows or columns of control points. It is worthwhile to point out that this method is inherently very unstable. However, it is often used in practice even though it should be avoided.

Linear extension

The simplest form of extension is using the derivatives across the extension boundary to infer the new geometry, see Figure 4.

This method is known as linear extension. In our case we extend each row of control points linearly in the direction given by

$$\boldsymbol{v}_i = \boldsymbol{d}_{n,i} - \boldsymbol{d}_{n-1,i}.$$

Note that now one additional segment is being created, and hence we increase the number of control points in each row. Typically, we have the freedom to achieve C^1 continuity by appropriate scaling. If we define

$$\alpha = \frac{\hat{u} - u_{n+d+1}}{u_{n+d+1} - u_{n+d}},$$

then by setting

$$\boldsymbol{d}_{n+1,i} = \boldsymbol{d}_{n,i} + \alpha \boldsymbol{v}_i,$$

we have achieved C_1 continuity at $u^* = u_{n+d+1}$ as can readily be verified. If we denote as $\boldsymbol{D}_+$ the partial of $\boldsymbol{s}$ at u^* computed from the (new) right segment and with $\boldsymbol{D}_-$ the corresponding partial computed from the left segment, we derive:

$$\boldsymbol{D}_+ = \frac{\boldsymbol{d}_{n+1,i} - \boldsymbol{d}_{n,i}}{\hat{u} - u_{n+d+1}} = \frac{\boldsymbol{v}_i}{\hat{u} - u_{n+d+1}} \frac{\hat{u} - u_{n+d+1}}{u_{n+d+1} - u_{n+d}} = \frac{\boldsymbol{v}_i}{u_{n+d+1} - u_{n+d}} = \boldsymbol{D}_-.$$

The other control points are usually placed equidistantly on the tangent line. Again rational surfaces are treated in homogeneous space, and it is possible that negative weights are created. One can remedy this by inserting knots appropriately.

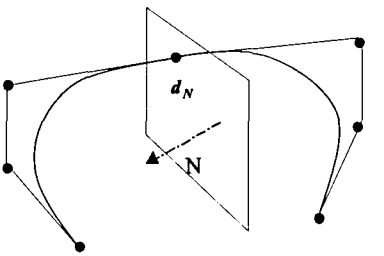


Fig. 5. Reflection of control points.

Reflection extension

This method has been introduced in [5]. The idea here is that more predictable results can be obtained by just mirroring the existing geometry across the normal plane. This is shown in Figure 5.

The basic operation of reflecting control points suffices when dealing with nonrational surfaces. The basic reflection operation can be formalized as follows: We again define $\boldsymbol{v}_i$ as in the case for linear extensions. Let us denote by $\hat{\boldsymbol{v}}_i$ the normalized vector. Then we have

$$\boldsymbol{d}_{n+j,i} = \boldsymbol{d}_{n-j,i} - 2 < \hat{\boldsymbol{v}}_i, \boldsymbol{d}_{n-j,i} - \boldsymbol{d}_{n,i} > \hat{\boldsymbol{v}}_i, \quad j = 1, \ldots, K,$$

where K is the number of new control points to be computed. It can be easily seen that G^1 continuity is preserved across the boundary $u = b$. A more subtle point to consider is continuity in v across a knot v_l with full multiplicity. If the original surface is C^1 continuous across v_l, we would like to ensure that the new part of the surface fulfills that condition as well. Control points generated by the simple extension equation above will *not* inherit C^1 continuity from the generating geometry. This is due to the fact that the normalization introduces a nonlinearity into the reflection formula. Only if we have the same reflection plane for the three rows affected is C^1 continuity across v_l achievable. G^1 continuity is achievable if one chooses the weight functions in the continuity equations appropriately. In the case of rational surfaces, more work is required in any case. The continuity conditions for adjacent rational patches are somewhat complex: Suitable weights and scalar values have to be computed by inserting the model space control points into the equations for G^1 continuity as stated in [5]. In addition, it is now not clear that G^1 continuity across a knot v_l with full multiplicity can be obtained. The authors in [5] ignored the complications arising by this configuration. Note that this problem is closely related to twist incompatibility issues when one is computing the new corner points.

Extending special surfaces

In most CAD systems, rational surfaces are only used to represent surfaces such as cones, cylinders, spheres and tori. In this case, one can create an

extension surface which is again a special patch by solving a simple system of equations without invoking the machinery of rational continuity conditions across boundaries: We make use of the fact that we are dealing with quadratic Bezier patches. So let us assume that we are given a biquadratic patch $\boldsymbol{p}(u,v)$ which again shall be extended across $u = b$. Let us denote the new patch by $\boldsymbol{q}$. The boundary control points are generated by simple reflection in model space. The weights are just copied, hence we guarantee positive weights. It remains to generate the point $\boldsymbol{q}_{11}$.We make use of the fact that the two end derivatives of the isoparametric curve formed by $\boldsymbol{q}_{01}$,$\boldsymbol{q}_{11}$, and $\boldsymbol{q}_{21}$ are the same up to a scale factor as the corresponding derivatives of the curve denoted by $\boldsymbol{p}_{01}$,$\boldsymbol{p}_{11}$, and $\boldsymbol{p}_{21}$. Furthermore, the scale factor λ is identical for both derivatives. This gives rise to a simple system of 6 equations in 4 unknowns: Let

$$\boldsymbol{D}_0 := \frac{\partial \boldsymbol{p}}{\partial u}(u = a, v)\frac{w_{01}}{w_{11}}\frac{u_4 - u_1}{2},$$
$$\boldsymbol{D}_1 := \frac{\partial \boldsymbol{p}}{\partial u}(u = b, v)\frac{w_{21}}{w_{11}}\frac{u_4 - u_1}{2}.$$

Then we obtain the equations

$$\lambda \boldsymbol{D}_0 - \boldsymbol{q}_{11} = -\boldsymbol{q}_{01},$$
$$\lambda \boldsymbol{D}_1 + \boldsymbol{q}_{11} = \boldsymbol{q}_{21}.$$

We know that a solution must exist, and by looking at the system, we can readily determine λ - for example by adding equation 4 to equation 1. Having determined λ, $\boldsymbol{q}_{11}$ follows trivially.

Hence we have presented an approach to compute the correct control points in the circular direction of a special surface such as cone, cylinder, sphere or torus.

Summary

We have presented the three commonly used extension methods. Let us summarize the advantages and disadvantages of each: Starting with the natural extension, its advantages are that maximal continuity is preserved and special surfaces retain their characteristic. The disadvantages are that depending on the original surface parameterization the results can be undesirable even when only extending a relatively small amount. Furthermore, since extension is performed in homogeneous space, weights can easily become negative even for special surfaces.

Linear extension is the most predictable method in that it resembles a ruled segment that joins the original surface with G^1 or C^1 continuity. However special surfaces are not preserved, and again we might produce negative weights when dealing with rational surfaces.

Reflection extensions yield positive weights for rational surfaces. The resulting surface is related to the original surface in a predictable fashion, at

least for a modest extension amount. It is possible to create G^2 continuous surfaces across the extension boundary. However, one might lose continuity in the other direction when knots with full multiplicity are present.

The combination scheme derived above combines reflection extension with a direct computation of inner Bézier points. This method has been developed especially for the extension of rational quadratic Bézier patches, and hence it preserves special surfaces without introducing negative weights.

Implementation issues

In order to implement a topology resolution system based on extensions, there are some other complicating factors to consider. First, the amount of extension has to be determined. In general this amount is given in parameter space. Depending on the parameterization, parameter space and model space might not correspond well. As a consequence, it is difficult to even predict the extension amount in model space without careful analysis of the given parameterization. Usually one needs to perform extensions in a loop by ways of callbacks. The process flow is as follows:

1) extend,

2) test for intersection,

3) if intersection found then process, else goto 1).

Of course, it is necessary to monitor this iteration. When finding an intersection curve requires extensions of significant amount, it is likely that the result is not acceptable. This is particularly true for shelling operations. When dealing with trimmed surfaces, the desired result might differ: In some cases it might be valid to extend the untrimmed surface; in other cases the trimming information must be preserved.

§4. Alternatives

We have seen that extension approaches are the weak link in topology resolution algorithms. Inherently, the problem of extrapolation is ill-defined. An alternative solution worth exploring is to reverse the process and to establish the desired intersection curve first. Given this intersection curve and optionally a tangent ribbon, one can construct a cubic Bézier patch blending between the intersection curve and the extension boundary and tangents constructing a G^1 continuous extension. Variations of this approach are possible as well: one might even prescribe four boundary curves and construct the extension patch by Coons blending techniques. If one needs to establish a vertex by performing multiple intersections, one could again establish intersection curves first, and then perform additional intersections to establish the vertex. To the authors knowledge, such an approach is currently not implemented in any commercial modeler.

§5. Conclusion

This survey article presented the methods used for extending curves and surfaces. Since they are all based on extrapolation, the algorithms are unstable and can lead to undesirable results. As a consequence, the topology cannot be resolved, and a valid solid cannot be produced. This leads to the failure of the entire local topology operation such as shelling, blending or drafting and loss of productivity for the end user who typically has to perform time-consuming steps to get the desired result. We have shown that all the algorithms have inherent weaknesses, and we have put forward a suggestion for alternative approaches. It is the authors hope that this paper motivates some much-needed further work in this area.

Acknowledgments. I wish to thank my former colleague Tim Strotman at SDRC for many fruitful discussions, and for helping me with his knowledge. Thanks to Sreedharan Vijayan for generating Figure 2.

References

1. Farin, G. E., *Curves and Surfaces for CAGD*, Academic Press, San Diego, 1996.
2. Hoffmann, C. M., *Geometric and Solid Modeling*, Morgan Kaufmann, San ateo, 1989.
3. Mäntylä, M., *An Introduction to Solid Modeling*, Computer Science Press, Rockville, 1988.
4. Requicha, A. G., and J. R. Rossignac, Solid modeling and beyond, IEEE Computer Graphics and Applications, **12**, Nr. 5, (1992), 31–44.
5. Shetty, S., and P. R. White, Curvature continuous extensions for rational B-Spline curves and surfaces, Computer Aided Design **23** (1991), 484–491.
6. Vida, J., R. Martin, and T. Varady, A survey of blending methods that use parametric surfaces, Computer Aided Design **26** 1994, 341–365.

Hans J. Wolters
Hewlett Packard Laboratories
Palo Alto, CA, USA
wolters@hpl.hp.com