ANNUAL REPORT
1 Oct 90 – 30 Sep 91

A Program of Continuing Research on
Representing, Manipulating, and Reasoning About
Physical Objects

Contractor:	Cornell University
Contract Number:	ONR N00014-89-J-1946
Modification Number:	P00001
R&T Project:	4333001-06
ACO Code:	N62927 ONR, Code: 1133
Cage Code:	4B578
Contract Amount:	$320,000
Effective Date of Contract:	May 1, 1989
Principal Investigator:	John E. Hopcroft
Telephone Number:	607/255-7316
Email Address:	jeh@cs.cornell.edu
Reporting Period:	1 Oct 90 – 30 Sep 91

Principal Investigator Name: John E. Hopcroft
PI Institution: Cornell University
PI Phone Number: 607/255-7316
PI E-mail Address: jeh@cs.cornell.edu
Contract Title: A Program of Continuing Research on Representing, Manipulating, and Reasoning About Physical Objects
Contract Number: ONR N00014-89-J-1946
Reporting Period: 1 Oct 90 – 30 Sep 91

1 Productivity Measures

Refereed papers submitted but not yet published: >4
Refereed papers published: 5
Unrefereed reports and articles: 14
Books or parts thereof submitted but not yet published: 1
Books or parts thereof published: 0
Patents filed but not yet granted: 0
Patents granted (include software copyrights): 0
Invited presentations: 10
Contributed presentations: 9
Honors received: 0
Prizes or awards received: 0
Promotions obtained: 0
Graduate students supported $\geq$ 25% of full time: 2
Post-docs supported $\geq$ 25% of full time: 0
Minorities supported: 0

Principal Investigator Name: John E. Hopcroft
PI Institution: Cornell University
PI Phone Number: 607/255-7316
PI E-mail Address: jeh@cs.cornell.edu
Contract Title: A Program of Continuing Research on Representing, Manipulating, and Reasoning About Physical Objects
Contract Number: ONR N00014-89-J-1946
Reporting Period: 1 Oct 90 – 30 Sep 91

2 Detailed Summary of Technical Progress

Objective

The Cornell Computer Science Modeling and Simulation Project was founded on the belief that electronic prototypes would have a significant impact on our nation's technological growth. From the project's inception, members of the research team have focused their attention on the investigation of fundamental computer science principles that underlie real-world representations.

Our overriding objective is to develop a science base that supports electronic simulation systems to be used in computer-aided design and engineering analysis. To this end, we are carrying out a broad-based research program to develop the underlying methodlology that supports representation, editing, and simulation of objects, tasks, and systems.

In our original proposal, we stressed three areas of research: solid modeler robustness, user-friendly modeling systems, and control of complex objects. Results in each of these areas are summarized below.

Solid Modeler Robustness

An algorithm is *robust* if its computer implementation never fails. We are investigating the design of robust geometric algorithms like those used in solid modelers and finite-element mesh generators. Most current algorithms assume infinite precision arithmetic and ignore the problems of finite precision. These algorithms fail because of roundoff error and imprecise input.

James Stewart, a graduate student supported by our project, has been particularly interested in algorithms for robust polyhedral intersection. A basic question that must be resolved when intersecting two polyhedra is where an edge of one polyhedron intersects the plane of a face of the other polyhedron; if the intersection point is *within* the face then it is common to both polyhedra and must be a vertex of the polyhedron of intersection.

In the previous reports the notion of an *approximate polygon* has been used to model polygons whose positions were uncertain. Using approximate polygons, Stewart has developed a robust algorithm which answers the point location question. This algorithm is fast and simple, and can handle uncertainty in both the location of the point and the polygon location. The algorithm can be used to build robust intersection algorithms, and hence, more robust solid modelers.

In related work, Mr. Stewart has concluded that formal correctness is nearly impossible to achieve in a robust polyhedral intersection algorithm, and that proving "local correctness" is the best alternative. A locally correct algorithm always succeeds when presented with localized special cases. For example, a locally correct intersection algorithm always succeeds when intersecting two corners, two edges, or two faces.

He has derived a set of *robustness rules* which, when incorporated in an intersection algorithm, cause the algorithm to be locally correct for intersections of two corners and intersections of two edges. Preliminary versions of this algorithm have been tested and found to be significantly more robust than earlier designs.

User-friendly Modeling Systems

Shape Control in Implicit Modeling

A useful geometric modeling system must provide a user with the ability to design and to manipulate three dimensional models represented by free-form surfaces. Such models are important when modeling mechanical parts, medical implants, and plastic flow in molds. Traditionally, free-form surfaces are built from parametric patches. Parametric patches have been successful for both design and rendering, but the manipulation of three dimensional models composed of parametric patches poses fundamental difficulties. For example, parametric patches are not closed under sweeping or convolution, and the intersection of a pair of parametric patches is extremely difficult to represent and evaluate.

One way to overcome the above difficulties is to build free-form surfaces from low-degree implicit patches. Implicit patches are closed under all common operations in geometric modeling, and the intersections of low-degree implicit patches can be computed efficiently. Implicit surfaces are also very useful in deriving blending surfaces. Recent research indicates that quadric and cubic implicit patches are versatile enough to build arbitrary three dimensional models.

A major reason for the popularity of parametric patches has been their good shape control properties. In recent work, Baining Guo, a graduate student supported by our project, has tackled the shape control issues of implicit patches. Using Bernstein-Bezier representation of polynomials, the shapes of implicit patches can be controlled by manipulating their control points. In designing free-form surfaces, one often encounters various shape requirements, such as a nice pattern of reflection lines or restrictions on the minimum radius of curvature. Among all the shape requirements, convexity is the most basic and the most frequently requested. Guo has shown how to manipulate the control points of a quadric patch or a cubic patch so that it becomes convex.

An Object Oriented Constraint Solver

Although computer programs for general constraint solving have existed for thirty years, few applications have been able to make use of them. Much of the reason for this is that the problems solved and the solutions delivered by these programs are not the problems encountered nor the solutions desired by most applications.

Michael Wilk, a graduate student supported by our project, has devised a constraint solver that has been designed to meet the needs of many applications, especially those with

an object-oriented design. Under this method, constraints are translated into procedures for achieving constraint satisfaction. Neither the constraints nor their procedural translations refer directly to an object's implementation; all object references are through the interfaces provided by classes. Because object classes guide the translation process, the method is applicable to objects of all data types.

The fundamental problem with typical constraint solvers is that they conform more to mathematical concepts than to any principles of programming, and certainly not to principles of object-oriented programming. A typical constraint solver will take a set of algebraic equations and return a set of assignments to variables. Even if the constraint solver provides a correct solution, we still might question whether the solver is suitable. One complaint is that typical constraint solvers violate object encapsulation. Encapsulation is first violated in the way the constraints are expressed. State variables are named explicitly, and to provide a constraint solver with a complete set of constraints each object is required to reveal the formulas used to maintain local consistency when information is stored redundantly. Encapsulation is also violated by solutions, which tell how to set individual state variables. These variables should not be manipulated directly; in circumventing an object's interface one might inadvertently set a state variable to an illegal value, or one might bypass a necessary side effect.

Wilk has recently developed the EQUATE constraint solver, designed specifically for use with object-oriented programming. EQUATE takes as input a constraint expressed as an equation, and produces as output a set of programs called *solutions*. Any solution executing successfully will cause the constraint to be satisfied. EQUATE itself does not satisfy a constraint; it translates a declarative constraint into procedural solutions. Constraints are recursively decomposed using definite-clause backward chaining, the search technique used in logic programming languages of which Prolog is the best-known example.

In EQUATE, constraints and their solutions respect the principle of object encapsulation. This should allow the incorporation of EQUATE into applications with relative ease. Because each class provides the rules for decomposing and solving constraints placed on instances of that class, EQUATE is especially useful for applications with heterogenous data.

Control of Complex Objects

Dexterous Manipulation

One of the great deficiencies of today's robots is their lack of flexibility. Most industrial robots are capable only of simple and repetitive tasks, such as spot welding or spray painting. There are two main reasons for this deficiency. First, typical end-effectors use a very simple two stick structure. Second, dexterous manipulation (manipulation by grippers with independently moving fingers) is not well understood. Effective techniques for dexterous manipulation would have application within a wide range of areas, including industrial assembly, decontamination of nuclear plants, and exploration of remote environments (*e.g.*, ocean bottoms or space).

We are developing a new strategy, *finger tracking*, for the autonomous manipulation of objects by multifinger robot hands. Most of the earlier efforts devoted to understanding dexterous manipulation have been devoted to grasping or to manipulation for task-specific problems. The finger tracking paradigm reorients an object with a series of rotations effected

by fine finger motions, in which the hand maintains contact with the object at all times. It is common for humans to reorient an object using "extra" fingers — those not needed for grasping. Finger tracking captures and formalizes these ideas.

Some of our results are summarized below:

- *The configuration space for a polyhedral object.* For the case of a polyhedral object held by a robot hand with four frictionless point contacts, we have obtained an algorithm to describe the configuration space as a manifold given by a closed form equation. We have analyzed the properties of the configuration space, and have shown that it is diffeomorphic to the rotation group $SO(3)$. Furthermore, we have shown that for this configuration space, the vertices of the polyhedron can move in a space-filling way. A consequence of this result is that the structure of the configuration space is quite complex, which makes finding finger tracking algorithms non-trivial.

- *Finger tracking for a polyhedral object.* Under the same assumptions as above, we have shown that the differential motion of the tracking finger is given by a 4×4 linear system. This surprising result is very feasible computationally, especially in the context of simulation.

- *Polygons in the plane.* Our newly developed framework for dexterous manipulation has been used to express earlier results for polygons; these results were originally obtained in a more *ad hoc* manner. Virtually the same algorithm used to determine the configuration space for polyhedra can be used to determine the configuration space for polygons.

- *Robustness for polygons.* The planar case has a number of geometric properties that we have been able to exploit in order to generate robust rotation algorithms. The uncertainty inherent in the real world makes robustness an important feature for any realistic manipulation algorithm. Our rotation algorithms are robust in the sense that some of the *a priori* knowledge requirements of the geometry of the object to be rotated and the necessity for precise calculations based on this geometry can be replaced by sensing. The result of our efforts is a condition on the geometry of the polygon to be rotated that guarantees robust rotations by an arbitrary rotation angle. We have shown that for convex polygons, the condition can be checked in $O(n)$ time, with $O(n \log n)$ preprocessing.

We are currently investigating the possibility of extending the robustness results from the planar, polygon case to the 3-dimensional, polyhedron case. We are also developing configuration space algorithms for 3-dimensional objects with curved faces. Another area in which we have made progress is the experimental verification of our results. We are using Newton, the simulator for rigid-body dynamics developed by our group, to verify our finger tracking algorithm for rotating polyhedra.

Walking using Least Constraint

We are concerned with controlling high degree of freedom mechanical systems which have to accomplish several simultaneous tasks. Such systems include robot arms, multi-fingered

hands, walking machines, mobile robots, and simulated mechanical systems used in computer animation. Such a system typically has redundant degrees of freedom for each task, but may have to accomplish a large number of tasks simultaneously. The system may also be autonomous and reactive, which means that a large amount of the "programming" will be done at execution time.

Dinesh Pai, a post-doc supported by our project, has proposed a framework called *Least Constraint* (abbreviated as LC) which we believe facilitates the expression of control programs for complex mechanical systems and is efficiently implementable as well. We have implemented an objected-oriented programming environment in Common Lisp based on this approach, and demonstrated its utility by programming a human-like walking machine to walk dynamically in three dimensions.

In the Least Constraint approach we do not specify the desired behavior such as walking in terms of trajectories, but rather we specify it more weakly as a collection of assertions to hold. These assertions are expressed as time varying inequality constraints in various domains. Thus, we attempt to capture the essential requirements of a task without imposing unnecessary structure. The constraints are solved at run time to produce control torques. Constraint satisfaction is performed using a fast iterative technique which takes advantage of functional decomposition and automatic differentiation.

The Least Constraint programming framework interacts closely with *Newton*, our rigid-body dynamics simulator. Programs are developed incrementally in Least Constraint and tested by immediately simulating the effects of the program on the system to be controlled.

Using this approach, we have developed a Least Constraint program for producing stable walking in a human-like walking machine model. The model has a torso and two legs with knees, and has mass properties and dimensions that are approximately that of a human. The model has fourteen degrees of freedom, of which eight are actuated. The walking task is factored into several subtasks such as foot placement for dynamic balance, ground clearance during leg swing, leg collision avoidance, support and orientation control of torso, and several others. Each subtask is expressed using inequality constraints. The various constraints are coordinated using a finite state automaton.

The program has been successful in generating stable dynamic walking in three dimensions, with various gaits. For example, by changing a few parameters, we have been able to make the simulated machine walk at different speeds, change the walking direction, and even walk with a slow, aperiodic gait. In current work, we are extending our algorithm for rough terrain walking.

Principal Investigator Name: John E. Hopcroft
PI Institution: Cornell University
PI Phone Number: 607/255-7316
PI E-mail Address: jeh@cs.cornell.edu
Contract Title: A Program of Continuing Research on Representing, Manipulating, and Reasoning About Physical Objects
Contract Number: ONR N00014-89-J-1946
Reporting Period: 1 Oct 90 – 30 Sep 91

3 Lists of Publications, Presentations, and Reports

Publications

1. Automatic Surface Generation using Implicit Cubics, *Scientific Visualization of Physical Phenomena*, edited by N. M. Patrikalakis, Springer-Verlag. Baining Guo.

2. A Case Study of Flexible Object Manipulation, *The International Journal of Robotics Research*, 10:1, February 1991, 41–50. John E. Hopcroft, Joseph K. Kearney, Dean B. Krafft.

3. Change Propagation in Object Dependency Graphs, *Technology of Object-Oriented Languages and Systems, TOOLS 5, Proceedings of the Fifth International Conference*, Prentice Hall, July 1991, 233–247. Michael Wilk.

4. An Efficiently Computable Metric for Comparing Polygonal Shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:9, March 1991, 209–216. E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell.

5. Least Constraint: A Framework for the Control of Complex Mechanical Systems, *Proceedings of the American Control Conference*, American Automatic Control Council, 1991, 1615–1621. Dinesh K. Pai.

6. Planar Sliding with Dry Friction: Part 1. Limit Surface and Moment Friction, *Wear*, 143 (1991), 307–330. Suresh Goyal, Andy Ruina, and Jim Papadopoulos.

7. Planar Sliding with Dry Friction: Part 2. Dynamics of Motion, *Wear*, 143 (1991), 331–352. Suresh Goyal, Andy Ruina, and Jim Papadopoulos.

8. Programming Mechanical Simulations, *Proceedings of the Second Eurographics Workshop on Animation and Simulation*, September 1991, 223–243. J. K. Kearney, S. Hansen, and J. F. Cremer.

9. Rational Function Decomposition, *ISSAC '91, Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation*, ACM Press, July 1991, 1–6. Richard Zippel.

10. Shape Control in Implicit Modeling, *Graphics Interface: 1991 Proceedings*, Morgan Kaufmann Publishers, 1991, 230–235. Baining Guo.

Presentations

1. *Dynamic Walking: Programming, Control, and Simulation*, Computer Science Department, Rensselaer Polytechnic Institute, January 31, 1991. Dinesh Pai.

2. *Modeling and Simulation for Electronic Prototyping*, DARPA Manufacturing Program PI Meeting, Salt Lake City, February 5, 1991. John Hopcroft.

3. *Symbolic/Numeric Techniques in Modeling and Simulation*, DARPA Manufacturing Program PI Meeting, Salt Lake City, February 6, 1991. Richard Zippel.

4. *Functional Decomposition*, Carnegie-Mellon, School of Computer Science, February 18, 1991. Richard Zippel.

5. *Dynamic Walking: Programming, Control, and Simulation*, Sibley School of Mechanical and Aerospace Engineering, Cornell University, February 19, 1991. Dinesh Pai.

6. *Symbolic/Numeric Techniques in Modeling and Simulation*, Carnegie-Mellon, School of Computer Science, February 20, 1991. Richard Zippel.

7. *Symbolic/Numeric Techniques in Modeling and Simulation*, DARPA Software Program PI Meeting, Providence, March 6, 1991. Richard Zippel.

8. *Programming Complex Mechanical Systems, with applications to Dynamic Walking*, Robotics Institute, Carnegie-Mellon University, March 8, 1991. Dinesh Pai.

9. *Integrating Symbolic and Numeric Computing*, New Directions in Computing Symposium, NASA Ames/University of Illinois at Chicago, April 8, 1991. Richard Zippel.

10. *Functional Decomposition*, University of Toronto, Dept. of Computer Science, April 25, 1991. Richard Zippel.

11. *A Framework for Dexterous Manipulation using Lie Algebras*, 2nd Algebraic Methodologies and Software Technology Conference, Iowa City, May 1991. Daniela Rus.

12. *The Role of Simulation in Science and Engineering*, Opening of the Theory Center, Cornell University, June 4, 1991. John Hopcroft.

13. *Shape Control in Implicit Modeling*, Graphics Interface '91, Calgary, Canada, June 3-7, 1991. Baining Guo.

14. *Entering the Information Age*, Seattle University, June 21, 1991. John Hopcroft.

15. *Robust Geometric Algorithms*, Seattle Pacific University, June 22, 1991. John Hopcroft.

16. *Least Constraint: A Framework for the Control of Complex Mechanical Systems*, American Control Conference, Boston, June 27, 1991. Dinesh Pai.

17. *Change Propagation in Object Dependency Graphs*, TOOLS USA '91 Conference, Santa Barbara, California, July 1991. Michael Wilk.

18. *Rational Function Decomposition*, International Symposium on Symbolic and Algebraic Computation, Bonn, Germany, July 1991. Richard Zippel.

19. *Symbolic Techniques in the Study of Turbulence*, IBM Europe Institute, Oberlech, Austria, August 1991. Richard Zippel.

20. *Programming Mechanical Simulations*, Second Eurographics Workshop on Animation and Simulation, Vienna, Austria, September 1991. J. K. Kearney.

Reports

1. *Masking Failures of Multidimensional Sensors*, Department of Computer Science Tech Report 91–1190, Cornell University, February 1991. L. Paul Chew and Keith Marzullo.

2. *Beyond Keyframing: An Algorithmic Approach to Animation*, Department of Computer Science Tech Report 91–1207, Cornell University, May 1991. A. James Stewart and James F. Cremer.

3. *Robust Point Location in Approximate Polygons*, Department of Computer Science Tech Report 91–1208, Cornell University, May 1991. A. James Stewart.

4. *Rational Function Decomposition*, Department of Computer Science Tech Report 91–1209, Cornell University, May 1991. Richard Zippel.

5. *Symbolic/Numeric Techniques in Modeling and Simulation*, Department of Computer Science Tech Report 91–1214, Cornell University, June 1991. Richard Zippel.

6. *Theory of R-functions and Applications: A Primer*, Department of Computer Science Tech Report 91–1219, Cornell University, July 1991. Vadim Shapiro.

7. *Boundary-based Separation for B-rep→CSG Conversion*, Department of Computer Science Tech Report 91–1222, Cornell University, July 1991. Vadim Shapiro and Donald L. Vossler.

8. *Modeling Arbitrary Smooth Objects with Algebraic Surfaces*, Ph.D Thesis, Department of Computer Science Tech Report 91–1226, Cornell University, August 1991. Baining Guo.

9. *The Theory and Practice of Robust Geometric Computation, or, How to Build Robust Solid Modelers*, Ph.D Thesis, Department of Computer Science Tech Report 91–1229, Cornell University, August 1991. A. James Stewart.

Principal Investigator Name: John E. Hopcroft
PI Institution: Cornell University
PI Phone Number: 607/255-7316
PI E-mail Address: jeh@cs.cornell.edu
Contract Title: A Program of Continuing Research on Representing, Manipulating, and Reasoning About Physical Objects
Contract Number: ONR N00014-89-J-1946
Reporting Period: 1 Oct 90 – 30 Sep 91

4 Transitions and DoD Interactions

- Researchers at Xerox and GE hope to apply our system to their own internal design and manufacturing problems. Two companies, GE and ORA (Odyssey Research Associates) have begun the initial work necessary to build simulator systems based on our ideas; GE for internal use, ORA for possible commercial development.
- We have continued our interaction with researchers at Purdue on the integration of geometric modeling and rigid-body dynamics within Newton, our rigid-body dynamics simulator.
- Newton, our rigid-body dynamics simulator, is being used in separate work at the University of Iowa and at the University of British Columbia to study the control of complex mechanisms.
- The current, experimental version of Weyl, our symbolic mathematics substrate, is being made available to colleagues around the country. At present, we are aware of Weyl's use by researchers at Harris Corporation, the Supercomputing Research Center, Xerox PARC, University of California at Berkeley, University of Illinois at Chicago, and the University of Massachusetts.
- At Cornell, Weyl has proved extremely useful in the investigation of turbulent boundary layers by John Lumley's group in the Department of Mechanical and Aerospace Engineering.
- Our work on mesh generators with provably good behavior has caught the interest of researchers at Xerox PARC, Purdue, and elsewhere, leading to additional techniques for provably good mesh generation.
- Researchers at Silicon Graphics Computing Systems are investigating the application of our meshing techniques to computer graphics with the goal of automatically converting complex graphics models into forms more appropriate for radiosity computation.

Principal Investigator Name: John E. Hopcroft
PI Institution: Cornell University
PI Phone Number: 607/255-7316
PI E-mail Address: jeh@cs.cornell.edu
Contract Title: A Program of Continuing Research on Representing, Manipulating, and Reasoning About Physical Objects
Contract Number: ONR N00014-89-J-1946
Reporting Period: 1 Oct 90 – 30 Sep 91

5 Software and Hardware Prototypes

5.1 Simlab

Over the past year we have been developing Simlab, a prototype software environment for automatically generating simulation and analysis programs. This prototype environment has been used to generate sample simulators for systems in rigid-body dynamics, electrical circuits, and particle dynamics. To build this environment, researchers at Cornell designed and implemented Weyl, a symbolic algebra substrate to provide for the necessary representation and manipulation of mathematical objects, developed a representation scheme for describing physical models and the associated laws for lumped parameter/ODE systems, implemented tools for deriving the appropriate set of equations from a given model (*i.e.*, a problem scene and the associated laws of physics), and built interfaces to numerical analysis libraries.

This prototype is an initial step in an ongoing effort to create a software environment in which an engineer can efficiently build simulators and other analysis tools for a variety of engineering disciplines. The environment's power comes from an integration of technologies, including geometric modeling, symbolic mathematics, object oriented programming, numerical analysis, and compilation/code generation. Using these technologies within an integrated environment for engineering analysis, a user can express engineering problems in the high-level, familiar and natural concepts of physics and mathematics rather than directly in a programming language such as Fortran. Programs expressed in this way can be more efficiently created, understood, and modified.

Two companies, GE and ORA (Odyssey Research Associates) have begun the initial work necessary to build simulator systems based on our ideas; GE for internal use, ORA for possible commercial development.

5.2 Weyl

An essential component of our activities is a symbolic mathematics substrate called Weyl. This substrate allows one to manipulate symbolic quantities (*e.g.*, polynomials or elements of a finite field) and mathematical domains (*e.g.*, the ring of integers or the field of rational functions with integer coefficients) within a conventional Lisp environment. Weyl is organized in functorial manner so that one can build arbitrarily sophisticated algebraic objects; for example, there is a functor that, given a ring, returns the ring of polynomials in X over the argument ring, and another that returns its quotient field.

The current, experimental version of Weyl is being made available to colleagues around the country. At present, we are aware of Weyl's use by researchers at Harris Corporation, the Supercomputing Research Center, Xerox PARC, University of California at Berkeley, University of Illinois at Chicago, and the University of Massachusetts.

5.3 Automatic Mesh Generation

For several commonly-used solution techniques for partial differential equations, the first step is to divide the problem region into simply-shaped elements creating a mesh. Our research has led to the development of a mesh generator for 2-dimensional problems and for curved surfaces that is guaranteed (by mathematical proof) to exhibit the following properties: (1) internal and external boundaries are respected, (2) element shapes are guaranteed — all elements are triangles with angles between 30 and 90 degrees (with the exception of badly shaped elements that may be required by the specified boundary), and (3) element density can be controlled, producing small elements in "interesting" areas and large elements elsewhere. In addition, our mesh generator is fast, generating a constant density mesh in optimal linear time.

Our work on mesh generators with provably good behavior has caught the interest of researchers at Xerox PARC, Purdue, and elsewhere, leading to additional techniques for provably good mesh generation. Researchers at Silicon Graphics Computing Systems are investigating the application of our meshing techniques to computer graphics with the goal of automatically converting complex graphics models into forms more appropriate for radiosity computation. More recently, researchers at Schlumberger have expressed interest in incorporating some of our mesh generation techniques in a CAD system they are developing.

5.4 Least Constraint

We are concerned with controlling high degree of freedom mechanical systems which have to accomplish several simultaneous tasks. Such systems include robot arms, multi-fingered hands, walking machines, mobile robots, and simulated mechanical systems used in computer animation. Such a system typically has redundant degrees of freedom for each task, but may have to accomplish a large number of tasks simultaneously. The system may also be autonomous and reactive, which means that a large amount of the "programming" will be done at execution time.

Dinesh Pai, a post-doc supported by our project, has proposed a framework called *Least Constraint* which we believe facilitates the expression of control programs for complex mechanical systems and is efficiently implementable as well. We have implemented an objected-oriented programming environment in Common Lisp based on this approach, and demonstrated its utility by programming a human-like walking machine to walk dynamically in three dimensions.

The Least Constraint programming framework interacts closely with *Newton*, our rigid-body dynamics simulator. Programs are developed incrementally in Least Constraint and tested by immediately simulating the effects of the program on the system to be controlled.