(12)

AD-A133 495

IMAGE UNDERSTANDING RESEARCH AND ITS APPLICATION TO CARTOGRAPHY AND COMPUTER-BASED ANALYSIS OF AERIAL IMAGERY

Final Report
Covering the Period Sept. 5, 1979 to Sept. 30, 1983

September 1983

Contract Amount:	$2,668,395
Effective Date:	September 5, 1979
Expiration Date:	September 30, 1983

By: Martin A. Fischler, Program Director, Vision
Principal Investigator, 415/859-5106

Andrew J. Hanson, Senior Computer Scientist
Image Understanding Testbed Coordinator, 415/859-4395

Artificial Intelligence Center
Computer Science and Technology Division

Prepared for:

Defense Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

Contract No. MDA903-79-C-0588
DARPA Order No. 3862
Program Code No. 61101E
SRI Project 1009

Approved for public release; distribution unlimited.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advance Research Projects Agency or the United States Government.

SRI International
333 Ravenswood Avenue
Menlo Park, California 94025
(415) 326-6200
Cable: SRI INTL MPK
TWX: 910-373-2046

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER SRI Project 1009 Final Report	2. GOVT ACCESSION NO. AD-A133 493	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Image Understanding Research and Its Application to Cartography and Computer-Based Analysis of Aerial Imagery		5. TYPE OF REPORT & PERIOD COVERED Final Report 9/5/79 to 9/30/83 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Martin A. Fischler Andrew J. Hanson		8. CONTRACT OR GRANT NUMBER(s) MDA903-79-C-0588
9. PERFORMING ORGANIZATION NAME AND ADDRESS SRI International 333 Ravenswood Avenue Menlo Park, California 94025		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 3862 Program Code No. 61101E
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE September 1983 13. NUMBER OF PAGES 108
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) DCASMA, San Francisco 1250 Bayhill Drive San Bruno, California 94066		15. SECURITY CLASS (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image Understanding, Computer Vision, DARPA/DMA Testbed, Automated Cartography, 3-D (Stereo) Compilation, Feature Extraction, Linear Delineation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The authors' ~~Our~~ principal objective in this research program has been to obtain solutions to fundamental problems in computer vision that have broad military relevance, particularly in the areas of cartography and photo interpretation. Now-completed research has been directed towards developing automated techniques for stereo-compilation, delineation of linear features, scene partitioning, image matching, and image to database correspondence. (CONTINUED ON NEXT PAGE)

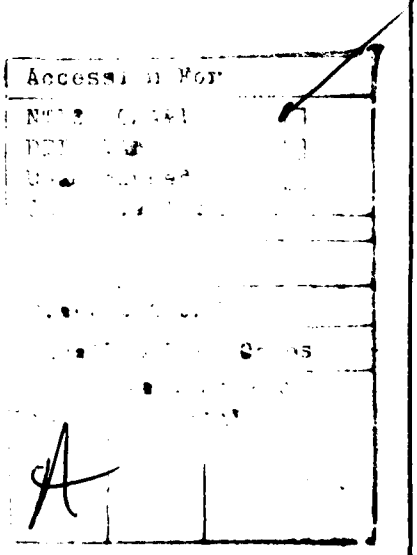
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT (continued)

In addition to our own research, we have designed and implemented an integrated testbed system that incorporates results of research produced throughout the Image Understanding community. This system provides a coherent framework for demonstration and evaluation of the accomplishments of DARPA's Image Understanding program, thereby facilitating transfer of this technology to appropriate military organizations.



CONTENTS

ACKNOWLEDGMENT

Contributors to the SRI Image Understanding Program include the following staff members: Stephen T. Barnard, Robert C. Bolles, Martin A. Fischler, Marsha Jo Hannah, Andrew J. Hanson, David L. Kashtan, Kenneth I. Laws, Alex P. Pentland, Lynn H. Quam, Grahame B. Smith, and Helen C. Wolf.

I INTRODUCTION

Research at SRI International under the DARPA Image Understanding Program was initiated to investigate ways in which diverse sources of knowledge might be brought to bear on the problem of analyzing and interpreting aerial images. An initial, exploratory phase of research identified various means for exploiting stored knowledge to support processing of aerial photographs for such military applications as cartography, intelligence, weapon guidance, and targeting. A key concept was the use of a generalized digital map to guide the process of image analysis. The results of this earlier work were integrated into an interactive computer system called "Hawkeye" [1].

Research subsequently focused on development of a program capable of expert performance in a specific task domain--road monitoring. The primary objective of this work was to build a computer system (called the Road Expert) that "understands" the nature of roads and road events; it is capable of performing such tasks as

* Finding roads in aerial imagery.
* Distinguishing vehicles on roads from shadows, signposts, road markings, etc.
* Comparing multiple images and symbolic information pertaining to the same road segment, and deciding whether significant changes have occurred.

The general approach, along with technical details of the Road Expert's components, are contained in the references [2-8]. We have integrated some of these separate components into a system that facilitates testing and evaluation, and have incorporated this system into the DARPA/DMA testbed.

Our more recent work on this now completed contract has addressed problems in two distinct topic areas:

* MACHINE VISION RESEARCH on selected problems in the areas of three-dimensional terrain understanding, linear-feature analysis, image partitioning, and image description and matching. This research program has been centered on the concept that image interpretation, except in the simplest situations, involves a form of reasoning ("perceptual reasoning") that is characterized by the need to integrate information from multiple sources, which are typically incommensurate and often erroneous or in conflict. We have developed a number of new techniques, even complete paradigms, for effecting the knowledge-integration task. These new techniques have been incorporated in the more focused efforts discussed in Section II, which address significant problems in scene analysis.

* THE DARPA/DMA IMAGE UNDERSTANDING TESTBED. Section III and Appendix A of this report describe the final status of the testbed. (Hanson and Fischler [9] describe the purpose and goals of the Testbed project in greater detail.)

II RESEARCH PROGRESS AND ACCOMPLISHMENTS

A. Three-Dimensional Compilation and Interpretation

The problem of stereo reconstruction is almost synonymous with the problem of machine vision--use of imaged data to (geometrically) model a sensed scene. A key concept in our approach [10-14] is the use of global physical and semantic constraints (e.g., sun location, vanishing points, edge detection and classification, skyline delineation) to resolve local ambiguities that defeat conventional stereo-matching techniques in mapping cultural or urban scenes, i.e., scenes that contain featureless areas and large numbers of occlusion edges, or scenes that are represented by widely separated or oblique views.

When a stereo pair of images is matched, even with the best possible use of available data (because of some of the problems mentioned above, e.g., occlusion and featureless areas), we generally can do no better than to compute a sparse depth map of the imaged scene. However, for many tasks a sparse depth map is inadequate. We want a complete model that portrays the scene's surfaces accurately. To achieve this goal, we must obtain the missing surface shape information from the texture and shading of the images of the stereo pair. We have made significant progress in understanding what is possible with respect to surface interpolation using scene shading. Pentland [15 and 16] and Smith [17] discuss some of our recent work in this research area.

B. Detection, Delineation, and Interpretation of Linear Features in Aerial Imagery

We have developed a system, called the "Road Expert," that can precisely delineate roads in both high- and low-resolution aerial imagery, classifying the visible objects that fall within the road boundaries [2-8]. A demonstration version of the Road Expert has been

installed on the DARPA/DMA testbed. We have investigated extensions of the above work to the problem of delineating other types of linear structures, such as rivers [18], and have recently made a significant advance towards developing a completely autonomous system for delineation of arbitrary linear structures [19].

C. Image Matching and Image-to-Database Correspondence

We have developed a new paradigm, called Random Sample Consensus (RANSAC), for fitting a model to data containing a significant percentage of gross errors, and have applied this paradigm to the solution of both matching/correspondence problems [20] and partitioning problems [21]. A RANSAC-based camera model solver has been developed and installed on the testbed. We expect that RANSAC will be equally applicable to a wide range of other model-based interpretation tasks and, under a separate contract, are investigating its use for recognizing and labeling known two- and three-dimensional scene features; this latter work even deals with unusual viewing or illumination conditions and partially occluded objects.

D. Image Partitioning, Intensity Modeling, and Material Identification

Our goal in this effort is to develop techniques for describing (partitioning and labeling) the material composition of a scene from available imagery. In order to recover information about actual surface reflectances and physical composition, the problem of intensity modeling must be addressed. We have developed methods for deriving absolute scene-intensity information without calibration data (such as a step wedge exposed on the image) based on knowing the identity of the material composition of the surfaces at a few points in the image--this capability is essential for the task of partitioning the image into labeled regions of given material types [10]. A new approach to the partitioning problem that is based on the concept of "fractal textures [22]" was developed, and work on this concept is being continued under separate funding.

III THE DARPA/DMA IMAGE UNDERSTANDING TESTBED

A distinct major focus of our program has been the establishment of an IU testbed system at SRI International. The DARPA/DMA Image Understanding Testbed constitutes a coherent body of research software running in a standard hardware environment. User documentation is available in four manuals:

* The DARPA/DMA Image Understanding Testbed User's Manual [23]
* The DARPA/DMA Image Understanding Testbed Programmer's Manual [24]
* The DARPA/DMA Image Understanding Testbed System Manager's Manual [25]
* Managing the IU Testbed Under EUNICE/VMS [26].

Software modules contributed by seven different institutions in the DARPA-sponsored research community may be demonstrated and examined on the testbed. Detailed evaluations have been carried out for three of the contributed systems; the results are reported in the following documents:

* The GHOUGH Generalized HOUGH Transform Package: Description and Evaluation [27]
* The PHOENIX Image Segmentation System: Description and Evaluation [28]
* The RELAX Image Relaxation System: Description and Evaluation [29].

A detailed report on the testbed is attached in Appendix A. Appendix B discusses our overall approach to the problem of evaluation. Appendix C consists of the RELAX evaluation document [29] cited above.

The Testbed is now established as a technology transfer tool that can be utilized by appropriate agencies to evaluate the applicability of the contributed scene analysis techniques. The testbed software system

and its utilities are being prepared for export to university researchers in the IU program as well as to other U.S. government agencies interested in establishing testbed copies. SRI has developed a testbed license agreement to help protect testbed contributors and restrict use of the software to appropriate research environments.

To support technology transfer requirements, exact copies of the entire system can be configured. SRI, under a separate contract, has completed installation of a testbed copy (hardware and software) at the U.S. Army Engineer Topographic Laboratories (ETL) at Fort Belvoir, and is continuing to support enhancement of the ETL research environment. A Lisp Machine will soon be added to the ETL configuration. SRI has also supplied Lisp Machines and Lisp Machine software to the DMAHTC and DMAAC branches of the Defense Mapping Agency. SRI has been closely involved in efforts to ensure that the upgrade of the DMA AFES/RWPF facilities to the VAX-11/780 CPU can incorporate the Image Understanding Testbed capabilities, as well as support the Lisp Machines.

We believe that the testbed is a very effective vehicle for conveying the results of IU research to government organizations concerned with image interpretation and automating cartographic tasks.

REFERENCES

1. H.G. Barrow, et al., "Interactive Aids for Cartography and Photo Interpretation: Progress Report, October 1977," in Proceedings: Image Understanding Workshop, pp. 111-127 (October 1977).

2. M.A. Fischler, et al., "Interactive Aids for Cartography and Photo Interpretation," Semiannual Technical Report, SRI Project 5300, SRI International, Menlo Park, California (October 1978 and May 1979).

3. L.H. Quam, "Road Tracking and Anomaly Detection," in Proceedings: Image Understanding Workshop, pp. 51-55 (May 1978).

4. R.C. Bolles, et al., "The SRI Road Expert: Image-to-Database Correspondence," in Proceedings: Image Understanding Workshop, pp. 163-174 (November 1978).

5. G.J. Agin, "Knowledge-Based Detection and Classification of Vehicles and Other Objects in Aerial Road Images," in Proceedings: Image Understanding Workshop, pp. 66-71 (April 1979).

6. R.C. Bolles, et al., "Automatic Determination of Image-to-Database Correspondence," in Proceedings Sixth IJCAI (1979).

7. M.A. Fischler, "The SRI Image Understanding Program," in Proceedings: Image Understanding Workshop (November 1979).

8. M.A. Fischler, J.M. Tenenbaum, and H.C. Wolf, "Detection of Roads and Linear Structures in Low-Resolution Aerial Imagery Using a Multisource Knowledge Integration Technique," Computer Graphics and Image Processing, Vol. 15(3), pp. 201-223 (March 1981).

9. A.J. Hanson and M.A. Fischler, "The DARPA/DMA Image Understanding Testbed," Proceedings: Image Understanding Workshop (September 1982).

10. M.A. Fischler, et.al., "Modeling and Using Physical Constraints in Scene Analysis," AAAI-82 (August 1982).

11. A.P. Witkin, "Intensity-Based Edge Classification," AAAI-82, pp. 36-41 (August 1982). (This publication won the best technical paper award for the conference.)

12. S.T. Barnard and M.A. Fischler, "Computational Stereo," ACM Computing Surveys, Vol. 14(4) (December 1982).

13. S.T. Barnard, "Methods for Interpreting Perspective Images," AI Journal (in press, 1983).

14. A.P. Pentland, "Depth of Scene from Depth of Field," Proceedings of the Image Understanding Workshop (September 1982).

15. A.P. Pentland, "Local Analysis of the Image: Limitations and Uses of Shading," Proceedings of the (IEEE) Workshop on Computer Vision: Representation and Control, Rindge, New Hampshire (August 1982).

16. A.P. Pentland, "Local Shading Analysis," IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), (to appear, March 1984).

17. G.B. Smith, "The Recovery of Surface Orientation From Image Irradiance," Proceedings of the Image Understanding Workshop (September 1982).

18. G.B. Smith, "Detection of Rivers in Low-Resolution Aerial Imagery," SRI Technical Note 244, SRI International, Menlo Park, California (June 1981).

19. M.A. Fischler and H.C. Wolf "Linear Delineation," " Proceedings of the Image Understanding Workshop (September 1982). Also presented at IEEE CVPR-83 (June 1983).

20. M.A. Fischler and R.C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *CACM, Vol. 24(6), pp.* 381-395 (June 1981).

21. R.C. Bolles and M.A. Fischler, "A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data," IJCAI-81, Vancouver, B.C., Canada, pp. 637-643 (August 1981).

22. A.P. Pentland, "Fractal Textures," IEEE CVPR-83 (June 1983).

23. A.J. Hanson, "The DARPA/DMA Image Understanding Testbed User's Manual," SRI Technical Note 277, SRI International, Menlo Park, California (February 1983).

24. K.I. Laws, "The DARPA/DMA Image Understanding Testbed Programmer's Manual," SRI Technical Note 298, SRI International, Menlo Park, California (February 1983).

25. A.J. Hanson, "The DARPA/DMA Image Understanding Testbed System Manager's Manual," SRI Technical Note 299, SRI International, Menlo Park, California (August 1983).

26. A.J. Hanson, "Managing the Image Understanding Testbed Under EUNICE/VMS," SRI Technical Note 300, SRI International, Menlo Park, California (August 1983).

27. K.I. Laws, "The GHOUGH Generalized HOUGH Transform Package: Description and Evaluation," SRI Technical Note 288, SRI International, Menlo Park, California (May 1983).

28. K.I. Laws, "The PHOENIX Image Segmentation System: Description and Evaluation," SRI Technical Note 289, SRI International, Menlo Park, California (May 1983).

29. K.I. Laws and G.B. Smith, "The RELAX Image Relaxation System: Description and Evaluation," SRI Technical Note 301, SRI International, Menlo Park, California (August 1983).

Appendix A

OVERVIEW OF THE IMAGE UNDERSTANDING TESTBED

OVERVIEW OF THE IMAGE UNDERSTANDING TESTBED

September 1983

Andrew J. Hanson, Senior Computer Scientist

Artificial Intelligence Center
SRI International

I INTRODUCTION

The Image Understanding Testbed is a system of hardware and software that is designed to facilitate the integration, testing, and evaluation of implemented research concepts in machine vision. The system was developed by the Artificial Intelligence Center of SRI International under the joint sponsorship of the Defense Advanced Research Projects Agency (DARPA) and the Defense Mapping Agency (DMA).

The primary purpose of the Image Understanding (IU) Testbed is to provide a means for transferring technology from the DARPA-sponsored IU research program to DMA and other organizations in the defense community.

The approach taken to achieve this purpose has two components:

* The establishment of a uniform environment that will be as compatible as possible with the environments of research centers at universities participating in the IU program. Thus, organizations obtaining copies of the testbed can receive new results of ongoing research as they become available.
* The acquisition, integration, testing, and evaluation of selected scene analysis techniques that represent mature examples of generic areas of research activity. These contributions from IU program participants will allow organizations with testbed copies to immediately begin investigating potential applications of IU technology to problems in automated cartography and other areas of scene analysis.

An important component of the DARPA IU research program is the development of image-understanding techniques that could be applied to automated cartography and military image interpretation tasks; this work forms the principal focus of the testbed project. A number of computer modules developed by participants in the Image Understanding program have been transported to the uniform testbed environment as a first step in the technology transfer process. These include systems written in UNIX C, MAINSAIL, and FRANZ LISP. Capabilities of the computer programs include segmentation, linear feature delineation, shape detection, stereo reconstruction, and rule-based recognition of classes of three-dimensional objects.

A. Documentation

The following documents relating to the IU testbed are now available as SRI technical notes:

* "The DARPA/DMA Image Understanding Testbed User's Manual" presents a user's view of the testbed. It outlines the general structure of the system and describes the use of major facilities.
* "The IU Testbed Programmer's Manual" collects UNIX-style "man" pages describing testbed programs, libraries, and files.
* "The DARPA/DMA Image Understanding Testbed System Manager's Manual" contains information relevant to system implementation and management issues.
* "Managing the IU Testbed under EUNICE/VMS" provides specific details for managers and users of systems that run the EUNICE/VMS emulation of the UNIX operating system.

The following reports evaluating major contributed software systems are also available:

* GHOUGH: "The GHOUGH Generalized Hough Transform Package: Description And Evaluation"
* PHOENIX: "The PHOENIX Image Segmentation Package: Description And Evaluation"
* RELAX: "The RELAX Image Relaxation System: Description And Evaluation"

Documentation describing the CMU-contributed graphics and picture-file access systems is provided in the following separate CMU documents:

* CMU002: "Grinnell Display Software Support"
* CMU003: "CMU Image Format and Paging System"
* CMU004: "Image File Naming Conventions."

B. Hardware Configuration

The principal elements of the IU testbed hardware configuration are a DEC VAX-11/780 central processing unit, with its peripherals, and several Symbolics Model 3600 Lisp Machines. The SRI testbed VAX is a four-megabyte system with one tape drive, four 300-MB disk drives, one 414-MB Winchester drive, and 32 teletype lines. The VAX interfaces directly to a variety of terminals. Graphics capabilities are provided by Grinnell and DeAnza display systems, both with 512 x 512 resolution and full color support. Several kinds of pointing devices, such as "mice" and digitizing tablets, are available. Other peripherals include a Versatec 11-inch printer/plotter with 200-point/inch resolution (which functions as a phototypesetter) and an Optronics C-4100 color image scanner with resolution selectable from 12.5 to 400 microns. The testbed system also supports an ARPANET network link with network address SRI-IU.

Each Lisp Machine has 2-MB of memory and a 180-MB disk drive. A 10-Mbit/second Ethernet network connects the Lisp Machines to one another and to the VAX. Color graphics systems and additional disk drives may eventually be added to enhance the capabilities of the testbed Lisp Machine environment.

C. Operating-System Software

The Image Understanding Testbed system may be run under either the UNIX* operating system or under the VAX/VMS** operating system. In

---

* UNIX is a trademark of Bell Laboratories.

** VAX/VMS is a trademark of the Digital Equipment Corporation.

principle, all testbed applications software can be run on either UNIX or VMS/EUNICE* systems, provided that appropriate system-specific hardware device drivers are available.

A "32V," or higher, UNIX license is required to operate the testbed under either system; in addition, a EUNICE license is needed to run the testbed under VAX/VMS.

The testbed currently uses the Berkeley VAX/UNIX 4.1c BSD system software distribution with support for the IP/TCP networking protocols; 4.2 BSD will be supported when it becomes generally available. UNIX device drivers are supplied by Berkeley for the Versatec printer/plotter and for ARPANET devices; a UNIX driver for the Grinnell display system has been provided by CMU. No Optronics scanner driver is available under UNIX at this time.

Under the VAX/VMS operating system, UNIX is emulated by the EUNICE system. This combination of operating-system support permits compatibility with both UNIX and other VMS/EUNICE environments. VMS device drivers are currently available for ARPANET devices, the Grinnell display system, the Versatec printer/plotter, and the Optronics image scanner. See the document "Managing the IU Testbed under EUNICE/VMS" for further details.

D. Languages

The principal high-level programming languages on the testbed VAX are UNIX C and FRANZ LISP. MAINSAIL, an ALGOL-like language, is available under both UNIX and VMS, but is currently used only on the SRI EUNICE/VMS testbed system. Other LISP dialects that may be used on the testbed include ISI VAX INTERLISP and MIT NIL VAX LISP. FORTRAN and PASCAL compilers are available under both UNIX and VMS, but are not used in any contributed software. On EUNICE/VMS systems, the DEC C-language compiler can be used instead of the UNIX C compiler for some applications; although the DEC C compiler generates exceptionally efficient code, substantial changes may be required to compile and run code written originally for a UNIX C system.

* EUNICE is a proprietary software product of SRI International.

Graphics functions on the testbed Grinnell display are fully supported in C; the testbed software is based on the CMU Grinnell graphics package. Supplementary testbed graphics capabilities are available for the DeAnza in MAINSAIL. FRANZ LISP and MAINSAIL programs may access the Grinnell by means of the C-language Grinnell graphics package.

Lisp Machine LISP is of course available on the Lisp Machines, which have now been integrated into the testbed system.

II CONTRIBUTED SOFTWARE

A. Overview of Applications Software Contributed to the Testbed

Besides SRI International, the institutions contributing software systems to the DARPA/DMA Image Understanding Testbed are Carnegie-Mellon University (CMU), the Massachusetts Institute of Technology (MIT), Stanford University, the University of Maryland, the University of Rochester, and the University of Southern California (USC). Modified or reimplemented versions of some routines have also been provided by a DARPA project at Hughes Aircraft Corporation.

Software modules integrated into the testbed include main programs, program systems, libraries of user utilities, graphics routines, and image access routines. Each of the designated testbed contributor sites has defined and delivered contributions to the testbed system. Among the research contributions are four modules from SRI and two from CMU; also running on the testbed are one contribution each from Rochester, Maryland, and USC, as well as a major system in FRANZ LISP from Stanford. MIT has provided a system in Lisp Machine LISP that runs on the testbed Lisp Machines. CMU has also furnished utilities, graphics, and picture access packages, while SRI has implemented an extended picture format and many additional utilities.

A summary of the currently operational research software contributions is given in Table I.

Table 1

SUMMARY OF TESTBED RESEARCH CONTRIBUTIONS

INSTITUTION	CONTRIBUTION	LANGUAGE
CMU	Picture access and display packages	C
	PHOENIX segmentation system	C
	Stereo/correlation system	C
MARYLAND	Relaxation package	C
MIT	Stereo reconstruction system	LISP MACHINE LISP
ROCHESTER	Generalized Hough transform system	C
SRI	Road expert	MAINSAIL
	RANSAC	MAINSAIL
	CAMDIST camera modeler	C
	SHOWDTM terrain map utility	C
STANFORD	ACRONYM 3-D model-based vision system	FRANZ LISP
USC	Linear-feature analysis	C

The following subsections summarize the status of each of the currently integrated contributions.

1. Carnegie-Mellon University Contributions

(1) CMU Grinnell Graphics and Image Manipulation Packages

* Date received: August 1981.
* Responsible party: David McKeown.
* Language: C (Berkeley UNIX) running on the VAX.
* Documentation: For complete documentation, see
  - CMU002: "Grinnell Display Software Support,"

- CMU003: "CMU Image Format and Paging System," and

- CMU004: "Image File Naming Conventions."

UNIX "man" entries providing high-level descriptions are available under the topics "cmuimglib," "gmrfrmlib," and "gmrlib." For testbed-based extensions to the CMU capabilities, see "dsplib," "frmlib," "imgfrmlib," "imglib," "imgnmelib," "piciolib," and "piclib."

* Description: These packages provide basic access to the functions of the Grinnell display system, as well as the capability of accessing image data files independently of the display system.

* Remarks: A number of minor modifications were needed to make the CMU package work with the SRI Grinnell configuration. The present code will support any CMU configuration or the SRI testbed configuration. The CMU image access package has also been integrated into the testbed environment; a new, extended testbed picture format has been implemented. Finally, there are several other general utilities of various sorts which have been supplied by CMU; see Section III.A.

(2) PHOENIX Segmentation Package

* Date received: December 1981.

* Responsible party: Steve Shafer.

* Language: C (Berkeley UNIX) running on the VAX.

* Usage: Invoke the command

```
phoenix inimage -o outimage -f featl [feat2 ...]
        [-i file | -I file]
        [-e] [-s] [-O file -r reg  -R reg ]
```

* Documentation: A "man" entry for PHOENIX is available under the topic "phoenix." Testbed documentation is provided in "The PHOENIX Image Segmentation System: Description And Evaluation." See also "Recursive Region Segmentation by Analysis of Histograms," a CMU preprint by S. Shafer and T. Kanade.

* Description: PHOENIX performs image segmentation by recursive region splitting. This segmentation package uses the Ohlander histogram-partitioning method to segment color imagery. Each pixel in the input image is assigned a segment identification label according to the image characteristics and the parameters selected. Segmentation is carried out hierarchically, with higher-level regions segmented into subregions. Segmentation ceases in a given

region when the program criteria for significance of the next level of segmentation have not been met.

* Remarks: This system has a sophisticated user interface and a checkpoint mechanism.

(3) Stereo Reconstruction and Correlation Package

* Date received: September 1981.
* Responsible party: Charles Thorpe.
* Language: C (Berkeley UNIX) running on the VAX.
* Usage: Invoke either of the two commands

```
correlate [-nqros v m t i ffilename -ffilename]

stereo
```

and answer the prompts for additional program input parameters.

* Documentation: "man" entries for CORRELATE and STEREO are available under the topics "correlate" and "stereo." See also "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Ph.D. Thesis by Hans Moravec.
* Description: This is a C version of the Moravec correlation and stereo reconstruction package written originally in SAIL at Stanford. The package consists of two portions: CORRELATE selects a set of "interesting" points in one image, using the Moravec interest operator, then attempts to locate the corresponding points in a second image by using an efficient hierarchical correlation matcher; STEREO uses the same method as CORRELATE to find corresponding points in a series of up to 9 images, then employs the Moravec method to assign a stereo depth value and confidence level to each match point.
* Remarks: This package implements all the basic capabilities of the original Moravec SAIL system, plus a number of enhancements introduced by Charles Thorpe.

2. University of Maryland Contributions

Relaxation Package

* Date received: Final version received 9 July 1981.
* Responsible party: Bob Kirby (author: Russell Smith, revised by Joe Pallas).

* Language: C (Berkeley UNIX) running on the VAX.

* Usage: Invoke the command

relax

or invoke various elements of the package individually. The individual programs making up the system include:

defcom defnbr imgprb prbimg relax relaxpar setup.csh

* Documentation: A "man" entry for RELAX is available under the topic "relax." Testbed documentation is provided in "The RELAX Image Relaxation System: Description and Evaluation."

* Description: This relaxation package takes an initial set of probabilities that a pixel belongs to each of a set of classes and iteratively adjusts them according to the probabilities of neighboring pixels. Two options are provided: an additive Hummel-Zucker-Rosenfeld relaxation algorithm and a multiplicative Peleg relaxation algorithm. A utility is provided for generating a two-class set of probabilities based on the luminance values of an image; the inverse operation is available to generate a grey-scale image from the reassigned probabilities, so that the user may monitor the relaxation process visually.

* Remarks: A multiclass method of generating probability assignments corresponding to luminance values has been added for test and demonstration purposes.

3. MIT Contributions

Marr-Poggio-Grimson Stereo System

* Received: February 1983

* Responsible parties: Mike Brady, Eric Grimson, and Keith Nishihara.

* Language: Lisp Machine LISP.

* Documentation: Current documentation consists of comments in the programs themselves. Additional documentation is planned by MIT.

* Description: This system uses zero-crossing matches at several scales to compute disparity values between stereo pairs. Additional consistency checking is available as an option.

* Remarks: This system makes use of an extensive package of Lisp Machine vision utilities, some generated at MIT and some revised or newly developed at SRI. In particular,

routines for reading and writing 8-bit images in testbed format have been provided; images may be read and written on the local Lisp Machine file systems, or may be read and written across the local network to the testbed VAX. The testbed Lisp Machine utility systems have been modified for use with the Symbolics 3600 Lisp Machines and now run in that environment. Convolutions are currently done in software. To enhance performance, it would be desirable to have convolution hardware on the Lisp Machines.

4. University of Rochester Contributions

Hough Transform Package

* Date received: May 1981.
* Responsible parties: Dana Ballard and Bill Lampeter.
* Language: C (Berkeley UNIX) running on the VAX.
* Usage: Invoke the command

  ghough

  and answer the prompts for program input parameters.

* Documentation: A "man" entry for GHOUGH is available under the topic "ghough." Testbed documentation is available in "The GHOUGH Generalized Hough Transform Package: Description And Evaluation."
* Description: This program takes a geometric-shape template and attempts to find matching shapes in the image, using the generalized Hough transform technique. The matched shapes may differ in displacement, rotation, and scale from the supplied template. The most likely values of location, rotation angle, and scale are printed out and the reoriented templates are displayed over the image.
* Remarks: The CMU graphics package has been used as a basis for incorporating full interactive graphics into this system for both template generation and picture processing. Several improvements have been made in the user interface as well as in the efficiency of the code. The package was extended to handle multiple instances of an object.

5. SRI Contributions

(1) Road Expert

* Date received: January 1981.

* Responsible parties: Lynn Quam and Helen Wolf

* Language: MAINSAIL running under EUNICE on the VAX.

* Usage: While connected to the /iu/sri/road/cmd directory, start up the MAINSAIL system and invoke the TRKACQ module.

* Documentation: A "man" page is available under the topic "road," along with demonstration instructions in the user's manual.

* Description: This package acquires and tracks linear features, such as roads, in aerial imagery. Tracking is done automatically in imagery with a known ground truth data base. Once a road has been identified and tracked, a separate subsystem is available to analyze road surface anomalies and to assign them to such categories as vehicles, road surface markings, and shadows.

(2) RANSAC Image-to-Data-Base Correspondence Package

* Date received: January 1981.

* Responsible parties: Martin Fischler and Robert Bolles.

* Language: MAINSAIL running under EUNICE on the VAX.

* Usage: While connected to the /iu/vision directory, start up the MAINSAIL system and invoke the INTMOD module.

* Documentation: A "man" page is available under the topic "ransac," along with demonstration instructions in the user's manual.

* Description: This package selects a best fit to an array of control points that may possibly contain gross errors. If such errors are present, RANSAC offers significant improvements over least-squares fitting techniques. A typical application is to compute the camera model from a given set of landmarks in aerial imagery.

(3) CAMDIST Camera Model System

* Date received: March 1983

* Responsible party: Marsha Jo Hannah

* Language: C (Berkeley UNIX) running on the VAX

* Usage: Invoke the command

camdist [options]

with desired options.

* Documentation: A "man" entry is available under the topic "camdist."

* Description: CAMDIST provides a facility for performing a generalized least-squares solution for the relative position and orientation angles between two cameras, given a series of points in the two camera views, and/or for using this information to calculate the distances to these points. Wild points are automatically edited out. Errors are propagated from the image plane points through the camera model to derive errors for the assigned distances.

(4) SHOWDTM Terrain Model System

* Date received: February 1983

* Responsible party: Marsha Jo Hannah

* Language: C (Berkeley UNIX) running on the VAX

* Usage: Invoke the command

showdtm [options]

with desired options.

* Documentation: A "man" entry is available under the topic "showdtm."

* Description: This is an interactive program for displaying a digital terrain model and producing either a perspective grid plot or a perspective range image of a portion of a model. When invoked with no arguments, SHOWDTM will prompt for the name of a terrain model (an image in testbed format), then wait for commands. If the name of an image file is specified, the program will open that file, then wait for commands. If an initial command string is specified, the program will execute each of those commands, then wait for more.

6. Stanford University Contributions

ACRONYM System

* Date received: March 1982.

* Responsible parties: Tom Binford and Rod Brooks.

* Language: FRANZ LISP running on the VAX. An extensive macro package is used to preserve most of the original MACLISP code.

* Usage: While connected to the directory /iu/acronym/sys, invoke "acronym". Connect to the models directory using (chdir "../models), invoke (PARSE model-file-name), and proceed with the desired ACRONYM process.

* Documentation: Some basic instructions are contained in /iu/acronym/info and are accessible by invoking the command file info.com; this command starts up an EMACS INFO system with a special ACRONYM node. Other information is available in /iu/acronym/doc. A "man" entry for ACRONYM is available under the topic "acronym." See also "ACRONYM: The Facts," a partially completed Stanford University document by Rodney Brooks. A more complete set of documentation will eventually be supplied by the Hughes Aircraft ACRONYM-based vision project.

* Description: ACRONYM takes a scene that has been reduced to a set of two-dimensional ribbons and searches for instances of three-dimensional models that have been supplied to the system as data. This is a rule-based system that allows great flexibility in interpretation and scene-prediction. Models can also be defined in a very general manner by using generalized cones, constraints, and subclass definitions.

* Remarks: Reduction of an image to a list of ribbons must now be done by hand, starting with a corresponding file of line segments generated by a program such as the Nevatia-Babu line finder. While some test imagery is available with the ribbon reduction already carried out, the testbed ACRONYM system would profit from the addition of an automated ribbon-reduction module. Such a module has been promised by the Hughes Aircraft project.

7. University of Southern California Contributions

Nevatia-Babu Line Finder

* Date received: June 1981 (SAIL version); June 1982 (C version from Hughes Aircraft).

* Responsible parties: Ram Nevatia at USC; Julius Bogdanovich at Hughes Aircraft.

* Language: C (Berkeley UNIX) running on the VAX.

* Usage: Connect to the /iu/usc/tst directory and run the following programs in sequence:

```
../bin/convolve
../bin/thrin
../bin/psmaker
../bin/linkseg
```

The output on seg.dat may be put into a device-independent display format by invoking ../bin/segdisp; the testbed graphics utility can be used to show the resulting display file.

* Documentation: See the Hughes Aircraft document "'C' VERSION OF THE NEVATIA-BABU LINEFINDER." A brief "man" entry is available under the topic "line."
* Description: This package extracts linear features from an image and produces a data base of line segments. The testbed C version supports 5 x 5 convolution masks configured to identify edges oriented at 30-degree intervals. The edges are then linked together into chains and broken into straight-line segments.
* Remarks: The C version of this package lacks the parallel-line (APAR) and supersegment (SAP) extraction routines present in the SAIL version. It would be useful for purposes of comparison to have these capabilities available. Support for using a variety of convolution masks would also be desirable.

B. Demonstration, Test, and Evaluation of Testbed Modules

The final stage of the testbed project at SRI was the demonstration, testing, and evaluation of the contributed software modules. Our purpose here was twofold:

* To provide information that would be useful for assessing the relevance of software techniques represented in the testbed.
* To establish a model for evaluation of comparable IU software.

Each module supports a standard demonstration of its capabilities. The degree to which testing and evaluation can be carried out meaningfully depends on the flexibility of each individual program. Some can run on completely arbitrary images, while others require extensive supporting data that cannot be easily assembled for arbitrary images. Furthermore, some contributions have been extensively documented in existing literature, while others have required additional modifications and documentation regarding their operation in the testbed environment. Accordingly, we have divided the contributions into the following two general classes:

(1) DEMONSTRATION ONLY. Several major stand-alone systems need customized data bases to function correctly. When tools for construction of such data bases are not available, the modules will run only on a limited set of images, thus restricting the nature of the evaluation that can be carried out. Such systems are not appropriate for systematic evaluation over large numbers of images because of the operational difficulties of setting up the required contexts. These systems are available for demonstration on limited data sets. Documentation of these systems on the testbed includes at least a "man" page and demonstration instructions in the user's manual. Some have additional manuals or are discussed extensively in the literature. Implementation details are generally undocumented, so users concerned with implementation methods must examine the software directly.

(2) DETAILED EVALUATION. Several modules that can readily be exercised on a wide variety of imagery have been subjected to rigorous investigation. Thorough evaluation reports have been prepared that describe the parameters, performance, strengths, and weaknesses of each of these modules. The detailed evaluation reports include the following:

* General description of the module and its scientific context.
* Scientific principles of operation of the algorithm.
* Program user documentation.
* Performance, strengths, and weaknesses.
* Suggestions for modifications.
* References and bibliography.

The following table summarizes the evaluation status of each of the contributed testbed software modules.

Table 2

EVALUATION STATUS OF EACH CONTRIBUTION

	CONTRIBUTION	EVALUATION OR DEMONSTRATION
CMU	PHOENIX	DETAILED EVALUATION
	Stereo/Correlation	DEMONSTRATION
MARYLAND	Relaxation	DETAILED EVALUATION
MIT	Stereo (Lisp Machine)	DEMONSTRATION
ROCHESTER	Hough Transform	DETAILED EVALUATION
STANFORD	ACRONYM	DEMONSTRATION
SRI	Road Expert	DEMONSTRATION
	RANSAC	DEMONSTRATION
	CAMDIST	DEMONSTRATION
	SHOWDTM	DEMONSTRATION
USC	Linear Features	DEMONSTRATION

C. Summary of Evaluation Results

The following modules were evaluated in detail:

* GHOUGH generalized Hough transform shape-finding system.
* PHOENIX segmentation system.
* RELAX pixel-level relaxation system.

In the evaluation process, we attempted to uncover characteristics of each system that normally become obvious to the user only through extensive experimentation. Summaries of the reports are given below:

1. GHOUGH

GHOUGH uses the generalized Hough transform method to find instances of a predefined template shape in an image. It allows the location, scale, and angular rotation of the target object to be determined. The system has also been extended to detect multiple instances of the same shape in a single image.

The following templates were used in testing the program: a lake, a right angle, a circle, and an ellipse. Several interesting artifacts of the template parameterization were observed. An example was the quantization of template angles resulting from the use of discrete lattice points to compute the orientation of line segments in the template. Very dense templates generated excessive noise compared to sparser outlines because neighboring pixels were related only by angles that were multiples of 45 degrees. This significantly increases the observed noise in the estimated object parameters. Several variations of the implementation strategy have been noted that would reduce such effects.

Other significant characteristics of the algorithm were observed during attempts to locate multiple instances of circular or elliptical storage tanks in a variety of aerial imagery. A powerful feature of the Hough method is its ability to discern incomplete and occluded shapes. On the other hand, no single choice of parameters would serve to locate accurately each and every one of the circular tanks that are obvious to the human observer; the blurred nature of some of the photometry and other characteristics of the tanks (e.g., rounded tops and shadows) required that special choices of parameters and templates be made for detection of any individual tank. Thus GHOUGH was found to be very useful in detecting unique, photometrically distinguished or partial shapes, but needed higher-level information to make effective parameter choices when the available imagery was less distinctive.

2. PHOENIX

PHOENIX is an Ohlander-style segmentation package that uses histogram analysis to carry out a hierarchical segmentation of color imagery. Several options are available to control the number and type of the segmentation cuts performed on each histogram as well as to select criteria for determining the significance of the segmentation.

The user interface for PHOENIX is based on the CMU CI command driver, which allows a wide range of subroutines to be called in an interactive and user-controlled manner. Information about each segment of a processed image can be printed and/or displayed on the graphics system as desired. Switches and flags are available to control graphics and other output from the program. A particularly useful feature is a checkpoint system that can save the current state of a segmentation process and read it back in at a later date for more detailed examination or additional processing.

A number of fundamental properties of the PHOENIX system have been noted. The best performance is obtained for color imagery in which objects of interest have distinct colors and for which the histograms of one or more spectral bands have at least two distinct peaks. Significant region identification in deeper levels of the hierarchical process also relies on the existence of more than one distinct peak in the histograms of the parent regions. Textured monochrome images often lack these characteristics.

PHOENIX can be utilized to advantage on imagery to carry out color-based region identification if the image digitization has a rich histogram structure. Transformations of the color space may have significant effects in adapting PHOENIX to specific segmentation problems. Given appropriate original or transformed imagery, one can use the output of PHOENIX for higher-level tasks that require image segmentation information.

3. RELAX

RELAX is a package that supports both the Hummel-Zucker-Rosenfeld and the Peleg pixel-level relaxation algorithms. To use the relaxation technique, one first assigns an initial set of probability values to the image pixels. For image-enhancement applications, there is a utility that converts a photometric image into a matrix of probabilities. A set of compatibility coefficients is then computed to support the relaxation computations. Finally, a number of relaxation iterations are performed to yield a new set of probability assignments. For image-based problems, the inverse of the original conversion utility can be run to generate a displayable grey-scale image representing the computed probability values.

The various steps in the application of the RELAX package have been integrated into a flexible user system that is based on the CI command interpreter system. We note that, for demonstration purposes, two-category relaxations allow fairly straightforward conversion between the imagery and probability structures. The usefulness of relaxation is highly dependent on the mapping from the initial imagery to the probability domain. Thus it is difficult to evaluate relaxation methods meaningfully in an abstract sense.

The RELAX system's ability to improve noisy imagery and to facilitate the extraction of image information depends strongly on the nature of the initial data and the probability assignments. The most effective way to use this system would be first to identify a subarea containing only one object of interest against a bland background, then to run RELAX to improve the signal. Alternatively, one could use an application-dependent preprocessor to assign probabilities based on criteria more complex than the values of individual pixels. This system produces excellent results if sufficient information is available for a meaningful assignment of category probabilities to the pixels of the original image, but may result in undue amplification of noise areas if the probabilities and compatibility coefficients are not chosen judiciously.

III TRANSPORTABLE FEATURES OF THE TESTBED ENVIRONMENT

One of the objectives of the testbed program has been to lay the foundation for a system that could be transported to other similar research environments. This transportability would allow other sites to make use of existing testbed code without having to develop their own versions; it would also make it possible for other sites to carry out their own evaluations and improvements of basic testbed contributions to meet their specific needs.

These objectives have been largely met. Each contribution to the testbed system can be tested and demonstrated with minimal modifications on UNIX or EUNICE/VMS VAX systems with Grinnell display devices. Many utilities have been acquired from contributing sites or developed locally by testbed personnel. A new and general testbed image file format has been created that supports all of the image types we have found useful in integrating contributed software. A modified version of the CMU image access package supports all essential image retrieval and access functions.

There are also several desirable objectives that remain to be achieved at this time. For example, graphics and image display on the testbed are supported entirely by an extension of the CMU Grinnell display package. This is a large body of software whose existence allowed basic testbed demonstration and testing objectives to be met in a timely fashion. However, the package is manifestly device-dependent, so each application program carries with it the device dependence inherent in using the Grinnell display package. It would be desirable to adopt a uniform device-independent graphics standard to support the testbed demonstrations on whatever devices happen to be available at a particular site.

Another objective is the establishment of a standard set of utilities for registering multiple images to a ground truth data base. Some progress has been made in this direction by the SRI RANSAC system,

the CAMDIST camera calibration system, and by the CMU "Browse" system (which is not yet ready for transport to the testbed). Further systematization of such image generation data as time of day, lighting characteristics, photometric parameters, and camera characteristics would also be desirable. The systematic application of IU techniques to cartographic tasks can only achieve its full potential when such information is available for all imagery used as source data.

In the following subsections, we present a summary of the basic capabilities that are supported in the testbed system and are potentially transportable to copies of the testbed.

A. Utility Programs

Among the generally useful utility programs available on the testbed are the following:

(1) CI. This is a command interpreter contributed by CMU. It allows a variety of subroutines to be linked into a top-level command processor and invoked with arguments provided interactively by the user. Extensive help and utility facilities are supplied.

(2) ICP. This is a command interpreter for the C language contributed by SRI. It is very similar to CI, except that its treatment of arguments and local variables is more general. ICP, for example, is able to invoke system or user subroutines directly, while CI must have an argument-parsing interface written for each routine.

(3) DOC. This is a CMU utility for generating program documentation (UNIX "man" entries) without having to know details of the TROFF phototypesetting system. All information the program needs to generate a syntactically correct "man" entry can be supplied interactively.

(4) CONVERT. This program supports color transformations, e.g., from red-green-blue to Y-I-Q or hue-intensity-saturation spaces.

(5) INVERT. Inverts a matrix of picture data to put the top row at the bottom, etc. This program can be used as a template for writing more general geometric or photometric transformations.

(6) NORMALIZE. This CMU routine normalizes a grey-scale image to produce a new output image with desired compression or clipping. SRI modifications allow grey-scale stretching as well.

(7) REDUCE. This CMU routine extracts a subwindow of an image or rescales an image by an integer sampling factor.

(8) SHAPEUP. The original CMU routine bearing this name has been entirely rewritten to support conversions among many image formats.

(9) VIEW. This utility is a data file listing program, analogous to the UNIX "od" octal dump program. It displays files that contain integer or floating-point two-dimensional data arrays, and is particularly useful for viewing compatibility and probability files produced by the RELAX system.

B. User Interface Systems

The following systems permit useful information or features to be made available to users of the testbed:

(1) TESTBED DEMONSTRATION DIRECTORIES. Complete demonstration facilities have been set up in the testbed demonstration directory, /iu/testbed/demo. Each of the contributions is represented by a series of subdirectories supporting various informative demonstrations of program capabilities. Ground truth data for comparison with program output is also available in some cases. The command files supplied in the demonstration directories provide detailed examples of program invocation; from these examples a sophisticated user can deduce the fundamental operating procedures for each program. Detailed written documentation of program usage is available in the evaluation reports for selected contributions.

(2) VAX EMACS INFO. An INFO macro package has been developed at the SRI testbed to support an extended version of the TECO EMACS INFO system. This system is a chain-linked documentation reading and generation system that utilizes the basic window-oriented features of the EMACS editor to access, search, and display text information. On-line testbed documentation is available through the INFO system. This provides a well-structured and convenient mechanism for access to the on-line documentation of the system's functions and capabilities.

(3) LEDIT and LTAGS. Intercommunicating modified versions of of EMACS and FRANZ LISP have been implemented on the SRI EUNICE/VMS system to support Lisp-Machine-like capabilities for developing FRANZ LISP programs. LEDIT allows the user to copy any defined function from a FRANZ LISP image into an EMACS editor buffer, modify it, and then reload it into the FRANZ LISP process without

changing any other part of the FRANZ LISP environment. Files with many functions can be edited in EMACS and the functions of interest marked for loading when the user returns to FRANZ LISP. The LTAGS package works in concert with LEDIT in EMACS, allowing the user to display any desired function in his window for editing by simply giving the first few characters of the function name; the system automatically keeps track of which files contain which functions. The system service capabilities needed to support the intercommunications involved in LEDIT are not now available on UNIX; they therefore require the VMS operating system.

(4) ARGLIB. This is a set of utility routines for parsing program parameters and interrogating the user for additional values.

(5) PRINTERR error package. This is a testbed package that supports flexible and user-friendly reporting and handling of error conditions.

C. Picture Data Base System

The testbed Picture Data Base System (PICDBMS) is a FRANZLISP-based system that interacts with a directory of test imagery to allow the entry and retrieval of image characteristics from an image data file. Following the CMU picture file conventions, each image is assigned a named directory (e.g., /iu/tb/pic/chair) that contains the picture data (e.g., 4red.img, 4blue.img, 4green.img) along with collateral data files. PICDBMS contains utilities for creating or editing a "pic.dat" file in each picture directory. This data file, containing data formatted for easy LISP readability, includes picture descriptions, picture characteristics, and a list of data base keys. Typical data base keys that are currently supported include the labels listed in parentheses below:

* IMAGE TYPE AND MULTIPLICITY: (bw color stereo multiple)
* SCENE DOMAIN TYPE: (indoor cultural natural)
* CONTENT CHARACTERISTICS: (point linear area)
* VIEWPOINT: (aerial ground).

Other types of data can be supported as the need arises. Sets of images can be retrieved by asking for images corresponding to a set of keys;

both AND and OR conditions are supported in the data base key interrogation.

Additional facilities of PICDBMS include a browsing utility to display lists of images provided by the keyed data base retrieval subsystem. Images too large to fit on one Grinnell screen can have any desired subwindows displayed in sequence.

IV PLANS

The future of the Image Understanding Testbed program at SRI will be closely tied to the SRI IU research efforts, as well as to the evolving characteristics of testbed copy systems to be installed at ETL and potentially at other DMA sites. The general applicability and transportability of the IU programs and utilities will continue to be enhanced as a by-product of the emerging needs of our research efforts. We anticipate that the recent incorporation of Lisp Machines into the environment will result in a substantial movement toward LISP-based IU application programs.

The major shift in emphasis in the testbed environment at SRI will be from low-level image-processing code towards increasing reliance on rule-based expert systems to guide the selection of low-level processes, the parameters to be used, and the interactive interfaces between the computer system and the human analyst. We foresee development of a substantial capability for supporting expert systems that will make it easier to apply IU research results to the solution of cartographic problems.

Appendix B

ON THE EVALUATION OF SCENE ANALYSIS ALGORITHMS

On the Evaluation of Scene Analysis Algorithms

Kenneth I. Laws, Computer Scientist

Artificial Intelligence Center
SRI International

1. Introduction

This paper describes software evaluation methods developed at SRI International to evaluate contributions to the ARPA/DMA Image Understanding (IU) Testbed. Examples of evaluation results are also presented.

The primary purpose of the IU Testbed is to provide a means for transferring technology from the DARPA-sponsored IU research program to DMA and to other organizations in the defense community. The approach taken to achieve this purpose has two components:

- The establishment of a uniform environment as compatible as practical with the environments of research centers at universities participating in the IU research program. Thus, organizations obtaining copies of the Testbed can receive a continuing flow of new results derived from on-going research.
- The acquisition, integration, testing, and evaluation of selected scene analysis techniques that represent mature examples of generic areas of research activity. These contributions from participants in the IU research program will allow organizations with Testbed copies to begin the immediate exploration of applications of IU technology to problems in automated cartography and other areas of scene analysis.

Evaluation of contributed scene analysis techniques has thus been a major thrust of the Testbed effort. Development of the evaluation methodology has been a related goal. Software evaluation is difficult, and few independent evaluations of IU software have been published. Analysis of an algorithm alone, even if feasible, would neither guarantee correct implementation nor quantify performance on realistic problems. Simple tabulations of pixel classification errors (as in Yasnoff, *et al.* [1]) would not be meaningful for complex scene analysis tasks. Comparative evaluation using several algorithms or software packages on one set of test scenes (as in Ranade and Prewitt [2]) was not practical for testing single algorithms. We have chosen a more subjective approach based on: (1) careful analysis, (2) tests on simple and complex natural scenes, and (3) our own experience in image analysis. This is similar to the method advocated by Nagin, *et al.* [3].

In this paper I describe my experiences with the initial software evaluation efforts on the IU Testbed. I was specifically involved with the evaluation of the GHOUGH object detection system [4] from the University of Rochester, the PHOENIX segmentation system [5] from Carnegie-Mellon University (CMU), and the RELAX relaxation package [6] from the University of Maryland. Many other software packages have been contributed to the Testbed, but have not been as extensively evaluated.

2. Evaluation Purpose

There are many reasons for evaluating software packages. Managers, systems personnel, and users all have different perspectives and different requirements. These imply many different questions that must be answered by a thorough evaluation effort. Some of the major questions are:

- Acquisition — Should the software package be acquired and further evaluated for local applications? What are its capabilities? Can it be extended?
- Implementation — What operating system support is required? How much memory does the package need? How much time does it take to run? Does the implementation correspond to the documented algorithm? Does performance match theoretical predictions? How well is the code structured and commented? Is the documentation adequate?
- Application — Is the package suitable for a particular application? Is the user interface adequate? How does the package perform? Can it be integrated with other packages?

We have attempted to answer these questions in our evaluation reports. The first section of each report introduces the package at a management level, answering questions about the tasks for which the algorithm is suited. Subsequent sections are written for system implementers and for users. The final sections document performance on evaluation tasks and make suggestions for future improvements.

An evaluation effort may have subsidiary effects on the software, the Testbed, and the personnel involved:

- Adaptation — The evaluation effort has spurred sev-

eral authors to polish or document their software before releasing it to the IU Testbed. Several contributions had to be translated into the C language before submission; the software thus became available on new classes of systems. Such "packaging" can be a significant step in the life of a software system.

- Validation — The processes of translating, installing, and evaluating contributed software have often led to the discovery of programming bugs and occasionally bugs in the algorithms. Where bugs are not found, there is greater assurance that such bugs do not exist.
- Training — We, the evaluating personnel, had to learn to use the software and to understand the theory behind it, thus extending the knowledgeable user community. We have documented this understanding and are otherwise communicating it to others.
- Documentation — Submission of software for evaluation has often spurred initial documentation of the package. Any weakness in this documentation were brought to light as we learned to use the package. We have then filled in the gaps and have added any necessary overview, literature survey, operating instructions, performance examples, and suggestions for improvement.

  We have also placed notes to future implementers and users in the source code and in the on-line man page documenting the Testbed version of the contribution. (These man pages are included in the Testbed programmer's manual [7].) We believe that such channels of communication between users scattered in space and time are essential for the continued growth of the software.
- Augmentation — We generally had to modify the submitted code to use local graphics and user interface routines, to instrument the code with additional displays or printouts of internal variables, and to rewrite portions of the code to eliminate trivial restrictions or to make the package more efficient for particular tasks. The dividing line between evaluation and new development is not clear, but it is clear that the evaluation effort often leads to improvements in the software. The Testbed environment also had to grow to support the contributions, and many ideas from the contributions have been adapted for use in other software.

3. Evaluation Structure

The tasks involved in evaluating a contribution are reflected in the structure of the evaluation report. We have developed this structure for recording and communicating the results of our investigations.

The introduction to a report summarizes the nature of the reviewed software package, the computer languages or system facilities needed to support it, and the contributions of various people in designing, creating, and maintaining it.

The succeeding background section describes the package from a management viewpoint. Generally this is one of the last sections written because it requires knowledge gained from the entire evaluation effort. First there is a general description of the package, including its purpose, inputs, processing steps, and outputs. Then typical applications and usage scenarios are described, including preconditions and the domain of applicability, relation to preprocessor and postprocessor programs, applications that have been documented in the literature, and potential applications that we or other researchers have suggested.

The background section also describes potential extensions and related applications. Potential extensions are applications that might be feasible if the package were modified or extended, used in a nonstandard fashion, or incorporated as an element of a larger system. Related applications are generalizations or variants of the standard applications for which other techniques seem to be more appropriate.

A descriptive section then documents the algorithm in detail. We begin with its historical development to introduce vocabulary and to put the major technical issues into perspective. Literature references are cited to give credit where credit is due, to aid researchers in finding the full range of concepts that have been explored, and to provide managers and implementors with contacts for further inquiries. The section closes with a detailed statement of the algorithm, including further discussion of design options and references to the literature as appropriate.

The next section is a brief implementer's guide describing the structure of the contributed software and the Testbed locations of its source files, executable files, on-line documentation, and demonstration files. This is information needed to install and run the package or to modify and maintain it. We have included here a description of the SRI modifications to the contribution.

A program documentation section then serves as a users' guide to running the package and invoking all of the algorithm features. We have given instructions for both interactive and batch (or background) execution, including documentation of all command-line invocation options, interactive commands, controlling variables and flags, and status variables. Sometimes we have also found it necessary to give a detailed description of the program's execution phases, complementing the theoretical description of the algorithm in previous sections. This section of the evaluation report could be omitted in cases where existing documents provide adequate and unified documentation of the program.

Our report can now document the evaluation proper. We have divided the evaluation section into two parts: effects of parameter settings and performance statistics for representative tasks. A subjective summary may also be included.

The purpose, intended effect, and legal values for each parameter and control variable were specified in the last section. In this section we probe more deeply, determining the true effect of each parameter on system performance

and documenting interactions (either constraints or synergistic effects) with other parameters. The end result is a set of rules for setting the parameters in various processing situations. We also comment on the usefulness of the control features and give suggestions for improving them.

Next we document the performance of the system on selected scene analysis tasks. (Selection of the tasks is discussed later in this paper.) We describe the test protocols, including the input images and the parameter settings that we found optimal for the tasks. We present subjective and objective performance measures and summarize the apparent strengths and weaknesses of the algorithm and the software implementation.

Subjective trials are difficult to document. We ran hundreds of trials on dozens of images. Often a trial designed to investigate one effect would turn up something else as well. It is impractical to illustrate each of these findings in the final report. (Many are of the form "Note how the edge detector found this weak edge but missed that much stronger one.") We have therefore attempted to summarize our findings and present only the relevant information.

In the next report section, we suggest substantial modifications to the algorithm or the implementation. Some of these are of potential, but uncertain, immediate benefit and some are extensions into task areas far beyond those considered by the original author. We also mention known improvements to the contributing institution's continuing software development that have not been incorporated into the more stable Testbed version. Many suggestions are derived from the work of other researchers, in which case we supply the appropriate references. Other suggestions arise from our own evaluation effort.

Our evaluation report concludes with a summary of the major technical concepts and of the strengths and weaknesses of the contributed algorithm and software. Appendices may give further information about the task domain, the algorithm, or the software package that is too detailed for the main body of the report but is not readily available elsewhere.

4. Evaluation Methodology

We have focused our evaluation efforts on the topics of greatest utility. Issues of applicability and of parameter effects and interactions have been given highest priority; issues of resource utilization have been given lower priority, because they are dependent on the algorithm implementation and supporting hardware.

Some of the most difficult evaluation issues have to do with the theory behind the algorithm. We have attempted to summarize the theoretical basis of each contribution, but evaluation of the theory is generally impractical. The best we could do is to document other approaches to similar tasks and to note strengths and weaknesses of the algorithm as reported in the literature or found in our own work. For this reason we have included an extensive literature survey in each of our evaluation efforts.

Fair evaluation of a contribution required that we choose particular tasks for it to perform. Some delicacy was needed in making these choices. It would be unfair to evaluate the software on tasks for which the author considered it unsuited. It would also be pointless to use only tasks that had been well documented in the literature; the essence of evaluation is the learning of something new. We have tried to choose tasks that are well within the contribution's domain and yet of fundamental interest to automated cartography and scene analysis.

The GHOUGH object detection system came with instructions for finding a distinctive lake in an aerial scene. We chose the finding of circular objects in aerial scenes and right-angled corners in oblique scenes as additional tasks. The PHOENIX segmentation system came with a test case of segmenting an orange chair from a white background. We chose skyline analysis as a realistic task. The RELAX probabilistic relaxation system was set up for noise cleaning of an infrared image of a tank. We chose gradient edge detection and segmentation of vehicles from roads as additional tasks. In each case, the imagery was rich enough that performance on auxiliary problems (*e.g.*, nonpurposive segmentation) could be subjectively evaluated.

For each task, we selected suitable imagery, ran dozens of trials to establish optimal parameter settings, and documented the results. If little documentation and operating information came with the package, we spent much of our time learning and recording this information. If documentation was adequate and few parameters had to be explored, we were able to spend more time recording operating characteristics and performance statistics. Economic constraints limited the depth to which any task could be evaluated, but we were able to provide an adequate foundation for future researchers with specific problems.

The first step in evaluating any package was to get it working on a simple test image — usually an image provided by the author. This process occasionally took considerable effort; we chose to rewrite the entire I/O structure and user interface for one program, for instance. This integration effort was essential to the development of the Testbed, but did raise a thorny issue: to what extent should we fix perceived deficiencies and to what extent should we simply document them? One rule of thumb was that we would fix or extend the code in any manner required to carry out an evaluation on realistic tasks.

The next step was to test the software rigorously on one or more simple images. The idea was to become familiar with the workings of the program and with the options available to the user. This step also helped identify software bugs or misunderstandings about the intended functions of the program. We strongly recommend the use of generated or well understood problems as one phase of the evaluation effort.

Investigation of parameter interactions was one of the most difficult evaluation tasks. Analysis of simple imagery permitted us to concentrate on internal variables instead

of interactions with a complex environment. Even so, the "search space" of possible interactions was immense. The PHOENIX program, for example, has 14 major threshold values to control image segmentation, in addition to various control strategies and interaction options.

One could navigate this complexity by using an intelligent driver system to monitor thousands of runs, modifying parameter settings each time to optimize some performance criterion. While such an approach is feasible [8], it would have provided only a superficial understanding of why the identified parameter sets were optimal combinations. We chose instead to analyze the program structure, experiment with carefully chosen parameter values, and study the execution (as opposed to a single result) of each computer run. Often we had to disable features of the program in order to study one feature in isolation.

The final experimental step was to evaluate the software for realistic tasks on "natural" imagery. This proved to be exceedingly difficult because the space of input imagery was impossibly large. If a program could detect circular tanks in one image, for instance, would it be able to detect them at different image resolutions? With different levels of contrast and blur? With strong shadows and highlighting present? With occlusions, unusual edge alignments, or texture effects present? Would it be able to distinguish real targets (possibly camouflaged) from decoys and destroyed targets?

Such questions go beyond the scope of this initial evaluation effort. We tried our best, however, to get a "feel" for each program's capabilities. We varied pertinent imagery variables and carefully noted the effects. Anomalies were checked out by instrumenting the code or by experimentation on simple images. We believe that this intelligent experimentation is at least as useful as extensive statistical validation would be.

Unfortunately we were not able to devise rigorous performance metrics for tasks such as "target detection." We carefully tuned the analysis system for each problem and reported the best performance that could be obtained. (We tried to avoid tuning the system for each image, however. A single parameter set or operating procedure was developed for each task.) The results are necessarily subjective and would vary slightly for other tasks, other imagery, or other experimenters.

5. Evaluation Examples

It is difficult to convey the scope and variety of our evaluation results in a short paper. The PHOENIX report alone is more than 80 pages long, with 25 pages devoted to performance evaluation and suggestions for future development. I will illustrate the evaluation results by presenting short excerpts from the reports. Different reports will be used to illustrate different points about the nature of the evaluation effort.

5.1. Theory

We have documented the theoretical basis and the implementation of each contribution from several perspectives. We have provided theoretical justifications and mathematical notations where appropriate, and have then related this information to the parameters and commands of the software packages. Sometimes it was quite difficult to extract this information from the technical literature.

The RELAX system, for instance, could be regarded as a general method for local modification of constraint and compatibility information stored in the nodes of a rectilinear graph. The initial label probabilities at the nodes may be derived from image pixel intensities and the final label assignments may be mapped back to pixel intensities, but the iterative relaxation technique is independent of image-domain considerations. Much of the theoretical work on relaxation has abandoned the rectilinear image plane and has dealt with constraint relations on arbitrary graphs with varying numbers of neighbors for each node.

For the evaluation, we extracted the updating equations actually used in the software package and expressed them in terms common in the theoretical literature. The RELAX package includes both the Hummel-Zucker-Rosenfeld additive updating scheme and the Peleg multiplicative updating scheme. Here is part of our description of the former.

> The goal of the relaxation algorithm is to update the values of the probabilities associated with a node to reflect the compatibility of neighboring labels. The $(t+1)$ update of the kth label value is calculated from the previous time (t) update by
>
> $$q_i^{(t)}(\lambda_k) = \frac{1}{m}\sum_j \left\{ \sum_{\lambda'=\lambda_1}^{\lambda_n} r_{ij}(\lambda_k, \lambda') p_j^{(t)}(\lambda') \right\}$$
>
> $$p_i^{(t+1)}(\lambda_k) = \frac{p_i^{(t)}(\lambda_k)\left(1 + q_i^{(t)}(\lambda_k)\right)}{\sum_{\lambda=\lambda_1}^{\lambda_n} p_i^{(t)}(\lambda)\left(1 + q_i^{(t)}(\lambda)\right)}$$
>
> where j indexes the m neighbors of node i. $r_{ij}(\lambda_k, \lambda')$ is the compatibility coefficient for node i with label λ_k and neighboring node j having label λ'. $q_i(\lambda_k)$ can be though of as the assessment by the neighboring nodes that node i should be labeled λ_k, while $p_i(\lambda_k)$ is the assessment by node i as to its own label. These two assessments are combined to produce an updated probability, $p_i(\lambda_k)$.
>
> The compatibility coefficients may be negative if the labels are incompatible, positive if the labels are compatible, and zero if they are independent. While it is possible to define the compatibility coefficients in terms of conditional probabilities, it is overly restrictive to do so. The compatibility coefficients for the Hummel-Zucker-Rosenfeld rule are based on information theory; mutual information defines the compatibility coefficients and provides a mechanism for calculating them:

$$r_j(\lambda_k, \lambda_l) = \frac{1}{5} \ln \frac{w \sum_i p_i(\lambda_k) p_{i_j}(\lambda_l)}{\sum_i p_i(\lambda_k) \cdot \sum_i p_i(\lambda_l)}$$

where k and l range over the n labels, i ranges over all w nodes of the graph, and j specifies the particular neighbor (*e.g.*, upper-left) of the i-th node. For each node i, $r_{ij}(\lambda_k, \lambda_l)$ is equal to $r_j(\lambda_k, \lambda_l)$ clipped to be in the closed interval [-1,1].

In the report we have discussed the meaning of these terms and methods of estimating or setting the numeric values, as well as the effects of relaxation in various image analysis tasks.

5.2. Analyses

The following are a few results from the performance analyses on the PHOENIX and GHOUGH systems. Space limitations prevent a full description of all of the terms used, but the examples should give some feeling for the level of understanding that a thorough evaluation may require.

The PHOENIX segmenter is a moderately complex system with 14 user-settable variables that control the segmentation process itself. The original contribution came with very little guidance about setting these parameters to achieve reasonable segmentations. One of our evaluation tasks was the creation of such information.

We began by finding a set of parameter values that would segment simple scenes of large objects down to the level of major subregions. We called this a "moderate" segmentation. Then we developed a set of parameter values for very coarse segmentation using "strict" heuristics to disallow most potential region splits. Finally we developed a set of values for complete or overly permissive segmentation using "mild" threshold screening heuristics.

Each column in Table 1 lists one of these parameter sets. The user need only select the extent of segmentation desired and load the corresponding parameters. We thus reduced the 14 parameters to a manageable single decision that is relatively independent of the image content. Additional flexibility is possible by switching parameter sets during a segmentation run to control the fineness of segmentation within particular image regions.

It will occasionally be necessary for the user to deviate from the recommended parameter settings. To make this possible, we have evaluated each parameter individually. Here is part of the **maxmin** parameter description:

> **Maxmin** is the minimum acceptable ratio of apex height to higher shoulder for an interval in the histogram. Any interval failing this test is merged with the neighbor on the side of the higher shoulder. The test is then repeated on the combined interval. The overall effect on a set of cutpoints is to eliminate those that are on the sides or tops of major peaks.
>
> **Maxmin** is a powerful heuristic. With strict smoothing and all other heuristics disabled, **maxmin** alone is able to produce reasonable segmentations. It is even more powerful when combined with the area heuristics. With mild or moderate smoothing, **maxmin** passes clusters of cutpoints in the noise regions between major peaks. This is fine if the clusters can be thinned by the **absarea** and **relarea** heuristics, but a poor selection may be made if they are left for the **intsmax** heuristic.
>
> The problem here is that PHOENIX has no "quality" score for histogram valleys. It assumes that cutpoint bin height is an adequate measure, whereas width and depth relative to the neighboring peaks are also important. PHOENIX can only incorporate such knowledge by smoothing the histogram, and the amount of smoothing required depends on how separated the peaks are.

Parameter	Strict	Mod.	Mild
depth	4	10	20
splitmin	200	40	20
hsmooth	25	9	5
maxmin	300	160	130
absarea	100	30	5
relarea	10	2	1
height	50	20	10
absmin	2	10	30
intsmax	2	3	6
isetsmax	2	3	5
absscore	925	700	600
relscore	95	80	65
noise	50	10	5
retain	4	20	40

Table 1: PHOENIX Segmentation Parameter Sets

The next step in the PHOENIX evaluation was investigation of a skyline delineation task. One of the test images, *portland*, shows a city skyline against a cloudy sky. After describing segmentation performance on reduced versions of this and other images, we reported the following:

> A test sequence was run on the full-resolution (500x500) *portland* image. Strict and even moderate heuristics were unable to segment the image when only the red, green, and blue feature planes were used; it was necessary to use the mild heuristics. The best approach would be to start the segmentation with mild thresholds and then return to strict or moderate ones for segmenting the subregions. Instead, we avoided such special interference and ran the segmentation to completion using mild heuristics. The full run (which, with the 'v' flag set, generated 19,000 lines of printout) required 33 minutes of CPU time:

PHASE	REAL	CPU
Histogram	0:04:13	0:02:32
Interval	0:18:12	0:07:27
Threshold	0:10:00	0:03:47
Patch	0:03:51	0:03:30
Collect	0:38:12	0:14:04
Segmentation	1:18:15	0:32:34

The final segmentation into 1182 regions (including

nearly every window of every building) was much better than the original attempt, but still had difficulties distinguishing a glass-surfaced building from the sky that it reflected.

Later test runs showed even better performance when color transforms were used in addition to the three original color planes. Based on our experience with these tests, we were able to suggest operating procedures for the use of PHOENIX in similar tasks.

The evaluation of the GHOUGH object detection system was similar. Because GHOUGH had fewer parameters, we were able to spend more time analyzing system performance on realistic tasks. One thrust of this effort was to develop an understanding of specific operational characteristics, as in the paragraphs below.

The requirement of sharp edges does not imply that smooth, continuous object boundaries are required. The program is quite tolerant of noise in the outline and is able to find irregular, incomplete, or discontinuous shapes. The circle template, for instance, often responds to forest clearings, tree tops, road intersections, and curved embankments, as well as to square buildings and to image "hot spots." The irregularities in these image structures spread the vote cluster in the accumulator, but the local maximum may still be above the general noise level.

Shadow edges usually fit the requirement for strong, sharp edges. It is often easier to find a shadow than to find the object that cast it. This may be a useful cueing technique, but must be used carefully to avoid reporting objects at incorrect locations. A similar problem exists with high-resolution imagery: the position reported for a part of an object (*e.g.*, the circular top of a storage tank) may not correspond to the position of the whole object.

These characteristics mean that the program is best suited for three tasks: locating industrial parts in high-contrast imagery; counting numerous, obvious, similar objects such as storage tanks, barracks, or microscopic particles; and precisely positioning a template when an approximate location is cued by the user or by another system. Even for these applications, the program must be supervised and its output edited. Other applications will require further development of the technique.

Sometimes our results were quite unexpected, as when we found that increasing the number of points in the search template definition had no effect on execution time and could actually decrease target location accuracy. Execution time was unaffected because each edge in the image votes for only the best matching template edge (or set of edges), regardless of the number of similar or nearby edges in the template. Performance could be degraded because the template points were entered at discrete points on a Cartesian grid, and close spacing of the points caused severe quantization of the relationships between them.

We were able to quantify system performance on representative tasks. Some of the domain-independent equations are given below. (The terms are fully explained in the evaluation report.) We have attempted to base the formulas on important characteristics of the GHOUGH algorithm, although the coefficients had to be estimated empirically.

$$\textit{Edge time} = .00036(\textit{window points}) + .0053(\textit{edges found}) + .00019(\textit{accumulator entries}) + (\textit{additional paging time})$$

$$\textit{Analysis time} = 10^{-4}(\textit{search volume}) \times (.08 + 2.6\log(1 + \textit{accumulator density})) + (\textit{additional paging time}) + .0025(\textit{maxima found})$$

$$\textit{Maxima} = .023(\textit{search volume})^{0.8}(1 + \textit{accumulator density})^2 - 1$$

$$\textit{Noise} = 2.04(\textit{search volume})^{0.99}(1 + \textit{accumulator density})^{0.8} - 1$$

Such formulas would be very helpful in designing an improved version of GHOUGH. Even more exciting is the possibility of building an expert image analysis system that would include GHOUGH as a component. The knowledge base of such a system would record predictive formulas and other operating characteristics in a form that could be used by both humans and machines.

Some of the GHOUGH parameters are dependent on image content. These were very difficult to quantify, but we attempted to document the dependencies well enough that users could adapt our findings to their own imagery. The following is our discussion of GHOUGH performance as a function of the edge-detection threshold.

The number and density of edges detected in an image are sigmoid (s-shaped) functions of edge threshold similar to cumulative frequency histograms. GHOUGH operates best when 10% to 20% of the pixels are classified as edge points, although it will usually work well at any edge density above 6%. Some typical threshold values to achieve specified edge densities are:

Scene Type	6%	12%	25%	50%
Cloudy sky	42	35	28	20
Aerial terrain	160	120	80	40
Aerial target area	200	180	120	60
Low-angle urban	200	200	140	90
Forest cover	280	220	160	100
Aerial urban	720	600	480	340

In general it is better to use too low a threshold: this will increase chances of finding target edges while only slightly increasing noise level, and the edges found are likely to be the most reliable ones. The main drawback is that low thresholds increase the time required to fill the accumulator with votes. A reasonable starting guess is a threshold of 120.

As we experimented with the software packages, we noted a great many characteristics that could be improved. The preceding GHOUGH edge-threshold sensitivity, for instance, led us to suggest that an adaptive edge detector be used. Our suggestions have covered everything from the algorithm to the characteristics of larger systems that might incorporate these routines.

5.3. Summaries

We have also tried to summarize our findings, drawing on our experience with other image analysis systems. Each of the reports ends with such a summary. For the PHOENIX system, our observations included the following:

> The PHOENIX segmentation system is one of several existing systems for recursively segmenting digital images. Its major contributions are the optional use of multiple thresholds, spatial analysis for choosing between good features, and a sophisticated control interface. Some of the strengths and weaknesses of the PHOENIX algorithm are listed below.
>
> PHOENIX, like other region-based methods, always yields closed region boundaries. This is not true of edge-based feature extraction methods, with the possible exception of boundary following and zero-crossing detection. Closed boundaries are the essence of segmentation and greatly simplify certain classification and mensuration tasks.
>
> PHOENIX is a hierarchical or recursive segmenter, which means that even a partial segmentation may be useful. This can save a great deal of computation if efforts are concentrated on those regions where further segmentation is critical. If PHOENIX is to be driven to its limits, other methods of segmenting to small, homogeneous regions may be more economical.
>
> PHOENIX is relatively insensitive to noise. Thresholds are determined by the feature histograms, where noise tends to average out. This contrasts with edge-based methods, where the local image characteristics can be highly perturbed by noise.
>
> PHOENIX has no notion of boundary straightness or smoothness. This may be good or bad depending on the scene characteristics and the analysis task. It easily extracts large homogeneous regions that may be adjacent to detailed, irregular regions (*e.g.*, lakes adjacent to dock areas or sky above a city); such tasks can be difficult for edge-based segmenters.
>
> PHOENIX tends to miss small regions within large ones because they contribute so little to the composite histogram. It is thus poorly suited for detecting vehicles and small buildings in aerial scenes, although there may be ways to adapt it to this use. It also tends to misplace the boundary between a large region and a small one, thus obscuring roads, rivers, and other thin regions. Boundaries found by edge-based methods are less affected by distant scene properties.
>
> PHOENIX may also fail to detect even long and highly-visible boundaries between two similar regions if the region textures cause their histograms to overlap. Edge-based methods are better able to detect local variations at the boundary.
>
> Since perfect segmentation is undefined, PHOENIX must oversegment an image in order to find all region boundaries that may be of use to any higher-level process. It is left for a segmentation editing step to merge segments that have no usefulness for some particular purpose. Without having such a step, or indeed even a purpose, it is very difficult to evaluate the segmenter output.

6. Conclusions

Our evaluation efforts have documented a great many suggestions for improving the evaluated software. We have tried to be as quantitative and rigorous as possible, but the results are necessarily subjective. Often we have functioned as restaurant or theater critics do, reporting our impressions of the contributions. These informed opinions, combined with our more rigorous documentation, should provide a good basis for more specific evaluation efforts directed at particular task scenarios and production environments. Our evaluation reports and the SRI Testbed environment make the contributed programs available as benchmark systems and as research tools.

REFERENCES

1. W.A. Yasnoff, J.K. Mui, and J.W. Bacus, "Error Measures for Scene Segmentations," *Pattern Recognition*, Vol. 9, pp. 217-231, 1977.
2. S. Ranade and J.M.S. Prewitt, "A Comparison of Some Segmentation Algorithms for Cytology," *Proc. 5th Int. Jnt. Conf. on Pattern Recognition*, Miami Beach, FL, pp. 561-564, Dec. 1980.
3. P. Nagin, R. Kohler, A. Hanson, and E. Riseman, "Segmentation, Evaluation, and Natural Scenes," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Chicago, pp. 515-522, Aug. 1979.
4. K.I. Laws, *The GHOUGH Generalized Hough Transform Package: Description and Evaluation*, Technical Note, Artificial Intelligence Center, SRI International, in press.
5. K.I. Laws, *The PHOENIX Image Segmentation System: Description and Evaluation*, Technical Note, Artificial Intelligence Center, SRI International, in press.
6. K.I. Laws and G.B. Smith, *The RELAX Image Relaxation System: Description and Evaluation*, Technical Note, Artificial Intelligence Center, SRI International, in press.
7. K.I. Laws, *The DARPA/DMA Image Understanding Testbed Programmer's Manual*, Technical Note, Artificial Intelligence Center, SRI International, in press.

8. D.B. Lenat, W.R. Sutherland, and J. Gibbons, "Heuristic Search for New Microcircuit Structures: An Application of Artificial Intelligence," *The AI Magazine*, Vol. 3, No. 3, pp. 17-33, Summer 1982.

Appendix C

THE RELAX IMAGE RELAXATION SYSTEM: DESCRIPTION AND EVALUATION

SRI International

THE RELAX IMAGE RELAXATION SYSTEM: DESCRIPTION AND EVALUATION

Technical Note 301

August 1983

By:

Kenneth I. Laws, Computer Scientist
Grahame B. Smith, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

Program Development by:

Russell C. Smith
Joseph A. Pallas

The University of Maryland

SRI Project 1009

The work reported herein was supported by the Defense Advanced Research Projects Agency under Contract No. MDA903-79-C-0588.

SRI International

333 Ravenswood Ave. • Menlo Park, CA 94025
415 326-6200 • TWX: 910-373-2046 • Telex: 334-486

Foreword

The primary purpose of the Image Understanding (IU) Testbed is to provide a means for transferring technology from the DARPA-sponsored IU research program to DMA and other organizations in the defense community.

The approach taken to achieve this purpose has two components:

(1) The establishment of a uniform environment that will be as compatible as possible with the environments of research centers at universities participating in the IU program. Thus, organizations obtaining copies of the Testbed can receive a flow of new results derived from ongoing research.

(2) The acquisition, integration, testing, and evaluation of selected scene analysis techniques that represent mature examples of generic areas of research activity. These contributions from participants in the IU program will allow organizations with Testbed copies to immediately begin investigating potential applications of IU technology to problems in automated cartography and other areas of scene analysis.

The IU Testbed project was carried out under DARPA Contract No. MDA903-79-C-0588. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States government.

This report describes the RELAX relaxation package contributed by the University of Maryland and presents an evaluation of its characteristics and features.

Andrew J. Hanson
Testbed Coordinator
Artificial Intelligence Center
SRI International

Abstract

RELAX is a system of routines that modifies the probabilities associated with labels attached to the elements of a two-dimensional array. These modifications reflect the compatibility of each element's labels with those of its neighbors. The initial probability assignments are usually derived from local property values in the neighborhood of each pixel. The final assignments may be used for object detection or segmentation, or may be mapped back to image intensities to achieve noise suppression, enhancement, or segmentation.

The relaxation package was contributed to the ARPA/DMA Image Understanding Testbed at SRI by the University of Maryland. This report summarizes applications for which RELAX is suited, the history and nature of the algorithm, details of the Testbed implementation, the manner in which RELAX is invoked and controlled, the type of results that can be expected, and suggestions for further development.

Table of Contents

Section 1

Introduction

The RELAX package is an interactive system of routines for mapping digital image data into a probability network, modifying the probabilities to reflect local constraints, and mapping the information back to the luminance domain. It is currently configured for image enhancement and object detection, but has many other applications.

Code modules and test data for the RELAX system were provided by the Computer Vision Laboratory at the University of Maryland (UM). The UM relaxation routines are configured as a set of stand-alone programs collectively called GPSPAR: General-Purpose Software Package for Array Relaxation. This package was originally written in the C language by Russel C. Smith and Joseph A. Pallas at the University of Maryland.

The current RELAX program is a command interpreter that interactively invokes the GPSPAR routines. This version of RELAX was constructed for the ARPA/DMA Image Understanding Testbed at SRI International by Kenneth Laws. The underlying command interpreter is the CI subroutine provided to the Testbed by Carnegie-Mellon University (CMU).

Many of the user-interface and image access routines were also contributed by CMU. Particular credit is due to Steven Shafer for the CI command interpreter and related string manipulation routines, to David Smith for the image access software, and to David McKeown, assisted by Jerry Denlinger, Steve Clark, and Joe Mattis, for the Grinnell display software. Kenneth Laws at SRI adapted this C-language software for Testbed use and interfaced it with the University of Maryland contributions.

No changes were required in the relaxation algorithm itself. The information in this document should thus be considered supplementary to the material cited in the UM references.

This document includes both a user's guide to the RELAX system and an evaluation of the algorithm. Section 2 explains the nature and use of the system in the context of typical applications. Section 3 surveys the historical development of the technique and presents the current algorithms in detail. Section 4 describes the Testbed implementation of this package and suggests some possible improvements. Section 5 instructs the user in the mechanics of using the RELAX software. Section 6 documents the performance that may be expected in various circumstances and presents the results of evaluation tests. Section 7 outlines a number of suggestions for improving the algorithm. Section 8 is a very brief summary. Appendix A shows how to invoke the RELAX routines in the manner of the original GPSPAR package submitted by UM.

Section 2

Background

This section presents a management view of the RELAX program. The relaxation algorithm is briefly sketched. Typical applications and potential applications requiring further development of the algorithm are discussed, and related applications for which other algorithms are better suited are noted.

2.1. General Description

The RELAX package for digital image enhancement and analysis is based on a class of algorithms for iteratively modifying vectors of probability values associated with the pixels of a two-dimensional array. This competitive-cooperative relaxation process strengthens compatible relationships and suppresses incompatible ones.

The RELAX algorithms also illustrate methods of propagating global interpretations and constraints through a network by local updating of the node interpretations. Such operations show promise for implementation on parallel arrays of processors and other advanced architectures.

An extensive literature connects the basic relaxation methods with numerous application areas. Much of the literature discusses relaxation on arbitrary graph structures rather than rectilinear data grids. The RELAX package, however, is aimed specifically at image-based applications.

The user starts with an array of values or a digital image, typically a luminance image or the output of an image-processing operator. The value at each pixel (or a set of values from a neighborhood of each pixel) is converted to a vector of probabilities; each probability reflects the likelihood that the pixel should be assigned a particular semantic label. This conversion process depends on the user's goals and the pixel classes that are relevant to those goals. In some formulations of relaxation, usually using different rules for adjusting the vectors at each pixel, the vectors can be regarded as representing fuzzy-set memberships [Zadeh65, Kandel78] rather than probabilities. In this report, for simplicity, we shall regard the vectors as probability vectors and and call the vector-valued image a probability image.

Next the user selects a relaxation method, a neighborhood size, and a set of compatibility coefficients. The compatibility coefficients are typically generated automatically from the initial probability image. This will be discussed in more detail in Sections 3 and 5.

The user then initiates one or more "relaxation steps," adjusting the probability vector at each pixel in accordance with the compatibility relationships and the probability vectors at each of its neighboring pixels. The definition of "neighbor" is supplied by the user; it must be the same for each pixel.

The resulting probability vectors are typically mapped back to the luminance domain

so that the user may observe the effect of the relaxation. This is not strictly necessary; a halting criterion based on the probability domain may be employed. The final probability image may or may not be mapped back to a luminance image, depending on the needs of the user.

Relaxation is a philosophically attractive procedure that seeks a globally consistent interpretation through local processing. Relaxation is still in the early stages of development and needs further research to determine the nature and range of its future applications.

2.2. Typical Applications

The RELAX program may be used in any application requiring noise suppression or feature reinforcement. The results of an image operation, such as edge detection, can be "smoothed" and detection reinforced. These effects, useful in themselves, may be precursors to further analysis.

The RELAX package is primarily adapted to specific applications by the mapping functions that convert luminance images to probability images. Very few such mappings are currently available with the package. Those that have been provided are suitable for the following purposes:

- Requantization--Reduction of the number of gray levels in an image typically introduces visible false edges in areas of smooth gradient. Relaxation may be used to pull pixels near the quantization threshold into the next higher or lower gray level. This will reduce false contours and act as a segmentation technique if the relaxation tends to group pixels that are within the same imaged object. (Adding a random dither signal to the image prior to requantization would also reduce false contouring, but would degrade the image and any subsequent segmentation of it.)

- Histogram Sharpening--There have been several schemes for iteratively replacing pixel values by some function of neighboring values in order to sharpen the peaks of the image histogram [Rosenfeld78, Peleg78b, Bhanu82]. Repeated applications can be used to merge smaller histogram peaks into larger ones until only a set number remain. (This differs from requantization in that the resulting gray levels need not be equally spaced.) Histogram sharpening is sometimes used as a precursor to image segmentation or compression.

- Smoothing--Relaxation can be used to smooth image regions to reduce noise artifacts. The smoothing can be done without blurring region edges if adjacent regions are mapped fairly well into different *a priori* labels. (Edge-preserving smoothing without such conditions requires special-purpose techniques that test region homogeneity before applying the compatibility correction. A median filter works this way.)

- Edge Enhancement--Relaxation can be used to sharpen region boundaries while smoothing the interiors. (Here, too, special-purpose algorithms that include decision logic might have better success than a linear summation of compatibility constraints.) Relaxation can also be applied to a gradient image to enhance extended discontinuities and suppress noise spikes.

- Linear-Feature Enhancement—This is essentially the same as edge enhancement, although more sophisticated feature detectors and classification operators might be involved.

- Detection—It is a small step from enhancing a feature to detecting it. Relaxation can help by reinforcing detection of groups of similar pixels while suppressing detection of isolated noise points. Other methods may be more advantageous for detecting objects larger or smaller than a few pixels.

- Pixel Classification—The source class to which a pixel is assigned may be adjusted by using the classifications and arrangement of its neighbors. Iteration of this process can reduce the effect of texture on classification. The RELAX package supplied by the University of Maryland contained a demonstration of two-class segmentation by thresholding, using an infrared image of a military tank. Segmentation by pixel classification into multiple classes using relaxation is described by Eklundh *et al.* [Eklundh80]. The anomaly detection experiments documented in Section 6 are also related to pixel classification.

2.3. Potential Extensions

The following applications might be feasible if the RELAX package were modified, used in a nonstandard fashion, or integrated into a more sophisticated system.

- Clustering—Clusters of points in a metric space can be detected by allowing each point to "gravitate" toward its neighbors. This is the spatial analogue of histogram sharpening. It requires a graph-based relaxation algorithm instead of the image-based RELAX updating algorithm.

- Semantic Labeling—Relaxation can be used to derive consistent sets of names or interpretations for regions in a scene. This also requires a graph-based relaxation method. A similar application is the identification of mixed pixels, noise regions, and border slivers in segmented images.

Such applications have been described in numerous papers, and there have been numerous other applications of relaxation techniques to image processing [Rosenfeld77b, Rosenfeld82, Rosenfeld83].

2.4. Alternative Approaches

This section describes applications that are similar to RELAX applications, but which differ in some fundamental fashion. While the difficulties with applying RELAX might be overcome, other techniques would often be more appropriate.

- Noise Suppression—Despite the applicability of relaxation to smoothing, the updating algorithms in the RELAX package have no underlying model of image and noise characteristics. Image noise can be more effectively removed by filtering techniques based on the noise statistics.

- Restoration—Similarly, blur and other degradations are best removed by techniques that model the degradation process. The RELAX package can be used for limited classes of image enhancement, but usually at the cost of introducing less visible degradations elsewhere.

As a rule, relaxation methods work best in those applications requiring "gravitational" or "fluid flow" solutions, such as histogram sharpening or image smoothing. They might also be useful for enhancement applications that can be cast as "reverse fluid flow" problems. They generally do not work as well as model-based restoration or analysis methods when there exist underlying models of the scene and the image formation process.

Section 3

Description

This section presents the history of relaxation for image processing and a detailed statement of the algorithms used in the RELAX package. The historical information is intended to clarify the major issues in iterative image processing and to guide the reader to the relevant literature.

3.1. Historical Development

Relaxation methods are iterative procedures designed to seek adequate solutions to problems that defy analytic analysis. Relaxation advances toward a solution state (*e.g.*, a fully segmented image) step by step, instead of solving for the optimal fixed-point solution (as is common in the image restoration literature). Either the user stops the process or an automatic halting criterion is invoked when the remaining errors are sufficiently small. The operation is similar to hill climbing, combinatorial optimization [Kirkpatrick83], or stochastic approximation approaches.

Relaxation methods have long been used in physics and engineering, particularly in computational fluid dynamics, aerodynamics, and thermodynamics. Systems of ordinary and partial differential equations are commonly solved by approximate numerical methods. Some finite-element techniques propagate local constraints through a field in a single pass; other techniques are iterative.

Relaxation techniques may have entered the image-understanding literature through constraint satisfaction networks used to label line drawings [Guzman68, Waltz72, Haralick79]. The early procedures assumed all possible labelings for each adjacent line pair in the scene, then eliminated incompatible label pairs. Convergence was very rapid, but these methods had no mechanism for handling probabilities or uncertain evidence.

Constraint networks were later generalized to many image-understanding and expert-system applications [Montanari74, Hart77, Rosenfeld77b], particularly to the consistent labeling of scene regions [Yakimovsky73, Barrow76, Freuder76, Faugeras81]. This movement merged with the development of iterative techniques for texture segmentation and identification [Troy73], image region growing and merging [Brice70, Yakimovsky76, Zucker76], image smoothing and enhancement [Davis77a, Lev77], histogram modification [Rosenfeld77a], edge detection [Eberlein76, Schachter76] linear-feature enhancement [Riseman77, Zucker77, VanderBrug77], curve segmentation [Davis77b], shape matching [Davis77c], and other applications.

The result, generally called relaxation labeling, is a set of iterative probabilistic approaches that consolidate many applications in the above areas. Probabilistic labeling, continuous relaxation, and stochastic labeling are other names for these techniques. All involve the application of local constraints to vary the weights (or degrees of belief) of semantic labels attached to the nodes of a graph. Iteration of the local adjustments, either in sequence or in parallel, are presumed to drive the

network closer to a global optimum or to a fixed point of the relaxation (*i.e.*, a state unaffected by further iterations).

The original relaxation labeling algorithm was proposed by Hummel, Zucker, and Rosenfeld at the University of Maryland [Rosenfeld76, Hummel78] and was further developed by a number of other researchers [Peleg78a, Zucker78a, Zucker78b, Yamamoto79, Haralick80, Hummel80, O'Leary80]. This method makes use of additive updating formulas. A later approach, based on multiplicative updating, was also developed at the University of Maryland [Kirby80, Peleg80a].

Many researchers have offered evaluation and discussion of the limits of relaxation processing [Kirby80, Kitchen80, O'Leary80, Richards80, Fekete81, Diamond82, Nagin82, Haralick83]. Faugeras and Berthod have suggested an alternative relaxation labeling philosophy [Faugeras80a, Faugeras80b], and this in turn has been criticized [Hummel80].

The papers cited above generally discuss relaxation in the abstract, although numerous applications have been developed [Rosenfeld82, Rosenfeld83]. The original papers on the Hummel-Zucker-Rosenfeld and Peleg methods [Rosenfeld76, Peleg80a] still constitute a good introduction to the philosophy of relaxation.

3.2. Algorithm Description

While the RELAX program provides a stand-alone implementation of relaxation, there are other ways of implementing it. Often the relaxation algorithm is so integrated with other techniques that it would be difficult to isolate the "relaxation part" of a procedure. Relaxation is a philosophy; no one algorithm embodies its alternate formulations. The two procedures included in the RELAX program (Hummel-Zucker-Rosenfeld, or HZR, and Peleg) are representative of the algorithms used in most applications of relaxation.

3.2.1. General Approach

Most image-based relaxation procedures comprise the four stages listed below; in rare circumstances one of the stages may be skipped. The stages are essentially input, construction of the relaxation operator, relaxation *per se*, and output. Each stage significantly influences the effectiveness of the overall relaxation procedure.

The four stages of the relaxation process are as follows:

- *Image-to-Probability Mapping*

  An operator is applied to the image array to convert the luminance values (or local property values, *etc.*) to probability vectors. Each pixel is assigned a vector representing an arbitrary set of semantic labels. Numeric values of the vector elements may be probabilities, likelihoods, or other measures of belief in the applicability of the corresponding labels at that image point. It is this mapping that adapts the relaxation paradigm to a specific application. The RELAX package currently contains illustrative mapping functions for image smoothing and edge detection applications. The user will generally have to supply new mapping routines for these or other applications.

Description

- *Compatibility Computation*

  Coefficients of compatibility between labels on neighboring pixels are computed, usually, from the weighted frequencies of label adjacencies in the original probability image. These coefficients essentially define the local relaxation operations that will be applied to the probability image. The RELAX package contains one routine for estimating HZR coefficients and another for Peleg coefficients; the values may also be supplied manually.

- *Relaxation Updating*

  The compatibility coefficients and updating rule are used to modify the probability vector at each pixel in turn. The values in each vector are adjusted by a weighted sum or product of values at neighboring pixels to enhance compatible combinations and suppress incompatible ones. The user may call for one or more passes of the relaxation operator through the probability image; current methods do not have halting criteria to determine when the updating should terminate.

- *Probability-to-Image Mapping*

  Relaxation methods may be designed to derive one or more numbers per pixel; these may be image luminances, edge probabilities, object likelihoods, segmentation maps, or other interpretations. The user must supply a mapping procedure to convert the multivariate probability image to this form. Routines currently in the RELAX package can produce binary and gray-level luminance images useful for edge or object detection as well as for image enhancement.

We shall now present these stages in detail.

3.2.2. Image-to-Probability Mapping

Luminance images used as input to a relaxation procedure must first be converted into probability image form. The method supplied in the RELAX package *imgprb* command is described here. It is a linear mapping of image brightness to a vector of probability values representing our belief that the pixel belongs in particular luminance "level slices." This mapping might be suitable for image binarization or requantization, noise suppression, and object detection applications. It is provided only as an example and as a mechanism for verifying the correct operation of the relaxation updating software; other mapping functions will be needed for other applications.

Among the arguments to *imgprb* are the low and high gray-level values and the number of labels to use. Pixel values are clipped to lie within the specified range and are then mapped to vectors of probability values between 0.0 and 1.0. The mapping function depends on the number of labels, as discussed below.

For binary output, the label probabilities may be thought of as positive and negative support for the hypothesis that a bright object is the pixel source. These are represented by two floating-point numbers per pixel, $p[0]$ and $p[1]$, where the indices represent the two classes or pixel labels. The first value, $p[0]$, represents

our belief that the pixel is from a light "object" area; the second, $p[1]=1.0-p[0]$, our belief that it is from a dark background population. The lowest pixel value in the specified range thus maps to $p[0]=0.0$ and $p[1]=1.0$; the highest to $p[0]=1.0$ and $p[1]=0.0$. Linear interpolation is used for intermediate pixel values.

If the user requests more than two labels, the labels may be thought of as level slices and the vector elements as probabilities that a pixel should be assigned to one of these levels. The lowest pixel value maps to $p[0]=1.0$, the highest to $p[$*maximum label*$]=1.0$. Linear interpolation between the bracketing labels is used for intermediate pixel values. At most two labels will have nonzero probabilities with this conversion scheme.

3.2.3. Compatibility Computation

A compatibility coefficient must be specified for each combination of a label at the central pixel with each label at each neighboring pixel. The values of the coefficients depend on the updating method to be used and also, as a rule, on the initial probability image data.

The neighborhood of a pixel is usually taken to be the set of pixels in a small surrounding square, with the size of the square selectable by the user. Neighborhoods restricted to only horizontal or vertical neighbors or that include nonadjacent pixels are sometimes required; the ***defnbr*** routine allows such arbitrary neighborhoods to be defined.

Compatibility coefficients used for the Hummel-Zucker-Rosenfeld relaxation scheme are different from those used for its Peleg counterpart. In the additive HZR scheme [Hummel78, Peleg78a, Yamamoto79], coefficients are negative if the labels are incompatible, zero if they are independent, and positive if the labels are compatible. Coefficient values are restricted to the range $[-1.0, +1.0]$. They typically range from -0.1 to about 0.5 when computed with the RELAX ***hcompat*** routine.

In the multiplicative Peleg scheme [Peleg80a, Haralick83], all the coefficients are nonnegative. Coefficients are less than unity if the labels are incompatible, unity if they are independent, and greater than unity if the labels are compatible. They typically range from near zero to about 5.0 when computed with the RELAX *pcompat* routine.

Compatibility is a loosely defined term, and no definition to date has been entirely satisfactory [Haralick83]. The HZR compatibility coefficients are based on information theory [Peleg78a, Yamamoto79], the corresponding Peleg coefficients on conditional probabilities [Peleg80a]. Both methods utilize the *a priori* probabilities of labels at an arbitrary pixel, as measured in the initial probability image, to estimate the compatibilities.

Suppose we have a graph whose nodes are each to be labeled with one of the possible labels $\lambda_1, \lambda_2, \ldots, \lambda_k, \ldots, \lambda_n$. Further suppose that some measurement associated with each node has allowed us to state the *a priori* probability of that node being labeled with each of the possible labels. For node i we have $p_i(\lambda_k)$, $k=1, \cdots, n$, as the probability that node i has label λ_k.

If we were to label the ith node with the label λ_k, and if the graph showed that the ith node and the jth node are adjacent, then we need to determine whether the label λ_k on the ith node is compatible with the labels on the jth node. We specify

this compatibility with the coefficient $r_{ij}(\lambda_k,\lambda_l)$, which states the compatibility of label λ_k on the ith node with the label λ_l on the jth node.

For the HZR scheme, compatibilities may be calculated from the quantity

$$r_j(\lambda_k,\lambda_l) = \frac{1}{5}\ln\left\{\frac{w\sum_i p_i(\lambda_k)p_{i_j}(\lambda_l)}{\sum_i p_i(\lambda_k)\cdot\sum_i p_i(\lambda_l)}\right\} \qquad \begin{matrix} k=1,\cdots,n \\ l=1,\cdots,n \end{matrix}$$

where i ranges over all w nodes of the graph and j specifies the particular neighbor of the ith node. For each node i, the compatibility with its jth neighbor is then

$$r_{ij}(\lambda_k,\lambda_l) = \begin{cases} -1 & \text{if} \quad r_j(\lambda_k,\lambda_l) < -1 \\ r_j(\lambda_k,\lambda_l) & \text{if } -1 \le r_j(\lambda_k,\lambda_l) \le +1 \\ +1 & \text{if } +1 < r_j(\lambda_k,\lambda_l) \end{cases}$$

Note that the RELAX package currently uses the same values of the compatibility coefficients for all values of i, *i.e.*, for each pixel postion to be updated.

For the Peleg scheme, the compatibilities may be calculated from the quantity

$$r_j(\lambda_k,\lambda_l) = \frac{w\sum_i p_i(\lambda_k)p_{i_j}(\lambda_l)}{\sum_i p_i(\lambda_k)\cdot\sum_i p_i(\lambda_l)} \qquad \begin{matrix} k=1,\cdots,n \\ l=1,\cdots,n \end{matrix}$$

where i ranges over all w nodes of the graph and j specifies the particular neighbor of the ith node. For each node i, the compatibility with its jth neighbor is then

$$r_{ij}(\lambda_k,\lambda_l) = r_j(\lambda_k,\lambda_l)$$

Note that these Peleg compatibilities are in the range $[0,w]$.

It may sometimes be desirable to use ensemble statistics to compute the compatibilities. Only experience with a particular application allows coefficients to be chosen rather than calculated by formula.

3.2.4. Updating Formulas

The relaxation updating computations can now be presented in more detail.

The goal of the relaxation algorithm is to update the values of the probabilities associated with a node so as to take into account the compatibility of neighboring labels. A number of different schemes have been proposed to do this updating. The earliest was the HZR scheme [Rosenfeld76], in which the $(t+1)$ update of the probability values is calculated from the (t) values by the following rule:

Description

For node i,

$$p_i^{(t+1)}(\lambda_k) = \frac{p_i^{(t)}(\lambda_k)[1+q_i^{(t)}(\lambda_k)]}{\sum_{\lambda=\lambda_1}^{\lambda_n} p_i^{(t)}(\lambda)[1+q_i^{(t)}(\lambda)]} \qquad k=1,\cdots,n$$

$$q_i^{(t)}(\lambda_k) = \frac{1}{m}\sum_j \left\{ \sum_{\lambda'=\lambda_1}^{\lambda_n} r_{ij}(\lambda_k,\lambda') p_j^{(t)}(\lambda') \right\} \qquad k=1,\cdots,n$$

where j indexes the m neighbors of node i, and $r_{ij}(\lambda_k,\lambda')$ is the compatibility coefficient for node i with label λ_k and neighboring node j with label λ'.

The Peleg relaxation scheme [Peleg80a], also included in this package, uses the updating rule

$$p_i^{(t+1)}(\lambda_k) = \frac{1}{m}\sum_j \frac{p_i^{(t)}(\lambda_k) q_{ij}^{(t)}(\lambda_k)}{\sum_{\lambda=\lambda_1}^{\lambda_n} p_i^{(t)}(\lambda) q_{ij}^{(t)}(\lambda)} \qquad k=1,\cdots,n$$

$$q_{ij}^{(t)}(\lambda_k) = \sum_{\lambda'=\lambda_1}^{\lambda_n} r_{ij}(\lambda_k,\lambda') p_j^{(t)}(\lambda') \qquad k=1,\cdots,n$$

where j indexes the m neighbors of node i. (Actually, a slightly more general form of the Peleg scheme is implemented in the RELAX system; see Section 5.2 for details.)

In both of these schemes, $q_i(\lambda_k)$ (or $q_{ij}(\lambda_k)$) can be thought of as the neighboring node's assessment (by node j in the case of $q_{ij}(\lambda_k)$) that node i should be labeled λ_k; $p_i(\lambda_k)$ is the assessment by node i that its own label should be λ_k. These two assessments are combined to produce an updated probability.

Other forms of the updating rule based on optimizing measures that are functions of terms like the above p's and q's, have been employed [Faugeras80a, Faugeras80b, Hummel80]. While the development of these forms rests on a firmer foundation than that of the rules above, these newer rules also have defects [O'Leary80, Peleg80b]. Substantial theoretical work needs to be done to comprehend the nature of relaxation and to lay the groundwork for eliminating rule deficiencies in the future.

The foregoing description has been couched in terms of the probability, likelihood, certainty, or favorability of assigning a particular label to a node. It is useful to think of the associated numeric value as the probability that the node has the label, but there are theoretical problems with this interpretation. We shall adopt the convenience of referring to such values as probabilities—but it should be kept in mind that, strictly speaking, this may not be correct.

Relaxation is an updating rule for improving the initial assignment of labels by enforcing compatibility with neighboring labels. Unfortunately, compatibility is a poorly-understood notion. One does not know when a rule will converge, or, if it does, what its rate of convergence will be [Zucker78a]. Nevertheless, relaxation has been successfully applied to a number of different problems. Relaxation

captures the ideas that we should be able to label objects so that they are compatible with their neighbors, and that we should be able to do this through local processing [Ullman79]. Relaxation requires a solid theoretical base [Hummel78, Hummel80] to define its domain of applicability.

3.2.5. Probability-to-Image Mapping

When the relaxation system is used for object cueing, the matrix of $p[0]$ values may be the desired output. In other cases, it may be desirable to map the probability vectors back to luminance gray levels so that the output can be displayed. This mapping is typically the inverse of that used to convert a luminance image to probability form. The RELAX package ***prbimg*** routine is the inverse of the ***imgprb*** mapping described earlier.

Inversion of the image-to-probability mapping is complicated by the fact that the "probability" vectors for a pixel are not always normalized and need not sum to 1.0. The inverse mapping function supplied by the University of Maryland uses different resolutions of this problem in the two-label and multilabel cases.

The ***prbimg*** routine will convert a two-label probability image to a gray-scale image whose values quantify the "strength of belief" in the $p[0]$ hypothesis represented by the stronger of the two probabilities. Thus, $p[0]$ values are converted directly to pixel values; $p[1]$ values, if stronger, are subtracted from 1.0 before conversion to pixel values. The gray-level interpolation inherent in this procedure produces an image quantized to an arbitrary number of gray levels, usually 256.

A multilabel probability image will be converted to a gray-scale image with values representing the label, or luminance-level slice, with the highest probability. Thus, a strongest $p[$***maximum label***$]$ maps to the highest pixel value and a strongest $p[0]$ maps to the lowest; intermediate labels map to intermediate gray levels. There is no interpolation between gray levels, so the output image is quantized to the number of labels used.

Section 4

Implementation

This section documents the SRI Testbed implementation of RELAX. It is intended as a guide for system maintainers and for programmers modifying the RELAX system. The terms used in this section are either defined elsewhere in this report or come from the supporting operating systems. The SRI Testbed uses the EUNICE operating system, which is a Berkeley UNIX[1] emulator for VAX computers using DEC's VMS operating system. All of the relaxation software will also run on a pure UNIX system.

The RELAX package is currently compiled as an interactive driver program. The driver invokes other programs for most of its work, although it does call subroutines directly to enable conversion between intensity and probability formats. The computational algorithm is very little changed from the original University of Maryland version, but the command interpreter, help system, and supporting image-access and display utilities are all new.

The main program and related files are in directory */iu/tb/src/relax*. Major subdirectories are

cvlhdrlib	- probability image subroutines;
defcom	- ***defcom*** source code;
defnbr	- ***defnbr*** source code;
demo	- shell script for the tank demo;
help	- help system text files;
imgprb	- ***imgprb*** source code;
prbimg	- ***prbimg*** source code;
relax	- ***relax*** main program source code;
relaxpar	- ***relaxpar*** source code (used by ***setup***);
src/hummel	- ***hcompat*** and ***hrelax*** source code;
src/peleg	- ***pcompat*** and ***prelax*** source code.

Compiled versions of these main programs are kept in */iu/tb/bin*. The Hummel and Peleg operators are not kept in compiled form, since they are intended to be customized for each application.

Cvlhdrlib is a library of subroutines for manipulating the floating-point probability files. (It is expected that someday this function will be absorbed by the Testbed image-accessing code.)

The ***imgprb*** and ***prbimg*** programs simply parse their command-line arguments and pass the information to subroutines. The RELAX driver likewise parses commands given to it and invokes the same subroutines. These subroutines are currently stored in directory */iu/tb/lib/visionlib*, specifically in the ***relaxlib*** subdirectory.

[1]UNIX is a trademark of Bell Laboratories.

Implementation

Source code and help files for the CI driver are in */iu/tb/lib/cilib*. For extensive documentation, type "man ci" or run "vtroff -man /iu/tb/man/man3/ci.3c". The CI driver uses command-line parsing routines in *cilib/cmuarglib* and in */iu/tb/lib/sublib/asklib*; both of these may someday be replaced by the Testbed argument-parsing routines in *sublib/arglib*.

Other utility routines contributed by CMU have been distributed to */iu/tb/lib/dsplib/gmrlib*, */iu/tb/lib/imglib*, and */iu/tb/lib/sublib*, and are documented for the **man** system in */iu/tb/man/man3*. Some of these have been modified or rewritten for the Testbed environment: the image access code, for instance, reads Testbed image headers in addition to CMU image headers. Other routines in these directories were developed at SRI.

To recompile one of the relaxation routines, *e.g.*, *imgprb*, just connect to the appropriate source directory and type "make". You may type "make -n" to see what will happen if you do this. Additional options are documented in the header of the makefile. To compile and install the entire system, run the *make* program in */iu/tb/src/relax*. For more flexible maintenance options, see the header sections of the corresponding *makefile* files.

The *hcompat* and *hrelax* programs and their Peleg equivalents are normally compiled interactively by using the *setup* command of the RELAX package. Implementation of this capability requires that the source file locations of these files be known to the RELAX program. This program must therefore be modified and recompiled any time the relaxation source files are moved.

RELAX demonstrations have been set up in subdirectories *relax* and *tank* of */iu/testbed/demo*. Just connect to the appropriate directory and run the *demo* command. Afterwards you may want to run the *cleanup* script stored in that directory to delete the relaxation output files.

The UM code represents an interesting style of programming and usage. Two points should be noted:

* Each routine is compiled as a separate program. Commands are passed to a command interpreter or to the UNIX shell to invoke the programs in the proper sequence. One benefit is that any routine can be altered without recompiling and linking the entire system. (Another possibility, not implemented here, is to pipe the routines together so that each feeds its output to the next. This would eliminate many of the intermediate files now being created by the system.)

* One of the steps in a relaxation sequence can be the construction and compilation of a special-purpose relaxation operation. This is currently done by the *setup* routine, which uses an *include* file built by *relaxpar* to tailor the *hcompat* and *hrelax* programs (or their Peleg equivalents) to the desired neighborhood definition. Such operators can run faster than general-purpose ones that use run-time evaluation of conditionals to control execution logic.

We have attempted to retain this style of programming while still packaging the relaxation system in a form similar to that of other major software systems on the Testbed. We have retained the concept of separate compilation so that the shell script in Appendix A will execute properly on the Testbed. Those who prefer such a system are free to

make use of it.

We have also written an interactive driver package, known as RELAX, for invoking the routines in a more structured environment; this allows for easy incorporation of customized syntax, argument defaulting, global status variables, help functions, and other "intelligent" session control features. (Very few such features are now implemented, but the possibilities can be seen in other Testbed software using the CI driver mechanism.)

Various problems were encountered in integrating the original package with the IU Testbed and in documenting the result. Sometimes it was easier to change the user interface slightly than to document an inconsistency. Among the changes made in the original system are the following:

- *Name Changes*

  The program to convert images to a probability format was originally named ***init***, and the program to convert them back again was named ***display***. We have changed these names to ***imgprb*** and ***prbimg***, respectively. The main programs now invoke subroutines to do most of the work; we have called these ***imgprbsub*** and ***prbimgsub***.

- *Conversion to Testbed Formats*

  The original ***imgprb*** and ***prbimg*** routines accepted images no wider than 512 pixels. We have removed this restriction. The pixels were also limited to 8 bits, or 256 gray levels. We have extended this range to the 36-bit pixels currently handled by the Testbed image access software. Testbed images of unusual pixel lengths, *e.g.*, 3 bits, are supported directly, as opposed to the UM practice of padding them into 8-bit fields with a "significant bits per pixel" specification to recover the dynamic range information.

- *Generalization to Multiple Labels*

  The original version of ***imgprb*** assumed that only two labels were to be used, although the rest of the package did accept more than two labels. We have extended the mapping algorithm as described in the previous section. The mapping to multiple labels was chosen to be the inverse of the mapping back from multiple labels that was already implemented in ***prbimg***.

  Prbimg accepted a "number of labels" argument, but then ignored it, since this information could be more reliably obtained from the header of the probability file. The demonstration shell script supplied with the original package erroneously omitted this argument in calls to the routine. This has been fixed by eliminating the interactive argument.

Implementation

- *Defaulted Arguments*

  The main programs are able to count the number of arguments passed to them and to substitute defaults for any missing arguments. We have retained this capability in the RELAX driver program. The invoked subroutines, however, must be supplied with a full set of arguments. We have adopted the convention that a negative minimum or maximum range specification passed to *imgprbsub* or *prbimgsub* will denote that the full dynamic range of the picture file is to be used. The user should specify non-negative values if stretching and clipping are desired.

- *Input Clipping*

  The *imgprb* routine originally stretched imagery to the specified gray-level range, but did not clip to this range. The mapping to a probability image was fine if the true image range was specified, but would generate probabilities that were negative or greater than unity for gray levels outside this range. The result of the inverse *prbimg* mapping on such a file was a sawtooth function. We have corrected this by clipping all probability values to the 0.0-1.0 range.

- *Word Size Conversion*

  The original package was written for a PDP-11 computer, for which the C language uses 16-bit integers. Our installation is on a VAX 11/780 that uses 32-bit integers. This difference is important in the writing and reading of *neighborhood and compatibility file headers, since* the I/O statements specified the number of bytes to be transferred. We have rewritten these sections to use the C *sizeof()* construct, which is guaranteed to be valid on any type of machine. The relaxation data files, however, cannot be transferred between machines with different integer sizes unless further standardization is implemented.

The need for additional changes became apparent during our software evaluation effort. Many of the needed improvements have to do with the command interpreter or the package philosophy rather than the relaxation algorithms. We suggest the following implementation changes:

- *Initialization Capability*

  The program should be able to run a startup file. This would allow the user to customize the package to his own preferences and tasks. Additional flexibility should be built into the command driver so that it could take advantage of initial profile information to set default neighborhood sizes and file names.

Implementation

- *Redefinition of Probabilities*

  The *imgprb* two-label mapping should be reversed to match its multilabel mapping interpretation. The *prbimg* inverse mapping should offer an option as to whether two-label probability files will be mapped to gray levels or to a binary output. Possibly a separate routine should be provided for each mapping and inverse mapping function instead of combining many functions in one routine.

- *Defcom Improvement*

  The *defcom* routine for interactively defining compatibility coefficients is very tedious to use. It is often easier to construct a file containing the coefficients and then pipe it into *defcom*, using the command interpreter's "<" facility for acquiring command input directly from a text file. If coefficients are provided in this manner and a neighborhood file is not specified, an inconsistency arises: the neighborhood size must be included at the start of the piped coefficients, even though they would not be needed if the coefficients were entered interactively. This should be changed so that the routine reading the coefficients does not expect to read the neighborhood size as well.

- *Run-Time Argument Passing*

  The command interpreter's "<" mechanism for invoking script files should accept arguments. The UNIX shell languages provide a good model for the type of argument macro expansion capability that is required.

- *Automatic Checkpointing*

  If automated iteration is ever added to the RELAX package, there should be some easy method of saving a checkpoint output every few iterations. Relaxation is such an expensive technique that a user should not have to start again from scratch if the system crashes or if processing has gone past the optimum point and begun to degrade the image.

- *File Name Flexibility*

  Part of the checkpointing problem is due to the current "hard-wiring" of the names *prb.img* and *compat.dat* into several of the routines. This causes the *hrelax* and *prelax* routines to overwrite the *prb.img* file, making it difficult to repeat an iteration (*e.g.*, with different parameters) or to recover after too many iterations. It is also difficult to remember exactly which iteration or processing sequence generated the current *prb.img* file. Although the UNIX/EUNICE hierarchical file system and the EUNICE multiple file-version facility alleviate some of these problems, the best solution is to allow arbitrary file names to be passed to the processing routines. An intelligent system for constructing default file names could also be helpful.

Implementation

- *Subroutine Implementation*

The RELAX routines are currently implemented as stand-alone main programs that can also be invoked from the RELAX driver. This differs from the subroutine-based implementation that is common to all other Testbed software. Although the main-program approach works, it is difficult to integrate with the rest of the Testbed. Our directory hierarchies and archiving techniques are based on libraries of subroutines rather than on clusters of main programs. It would be simpler to maintain a system that has a uniform programming philosophy.

As an example, consider the compilation of the *hrelax* routine. Ordinarily, each main program in the Testbed is accompanied by a *makefile* script that "remembers" all the *include* files and libraries needed in compiling the routine. If *hrelax* is to be created by the *setup.csh* script provided by UM, the compilation command must be in that script. If it is to be created by the RELAX program, the compilation command must be compiled into that code. Thus, changes in the structure or location of the *include* files and libraries must now be implemented by changes in several types of source code. While this is not difficult, it is certainly more trouble than updating a single type of file used consistently throughout the system.

The chief reason for using main programs rather than subroutines is that optimized relaxation operators are compiled for each specific task. This exchanges extra compilation time for reduced execution time--which seems reasonable, given the length of time currently required to run a relaxation step. Separately compiled subroutine modules could be linked into the driver package at run time, although this UNIX capability is not easily accessible or commonly used.

The shell languages do provide convenient mechanisms for sequencing programs, but they are better suited for batch execution than for interactive use. Furthermore, they are general-purpose and hence lack the focused environment, vocabulary, defaults, and help facilities that a dedicated command driver can offer. Command interpreters can invoke either main programs or subroutines, but are somewhat easier to write for the subroutine case.

Another benefit of subroutine-based implementation is control of interprocess communication. Programs invoked by shell scripts can communicate only via files. (Communication by means of shell environment variables is also possible, but not commonly done.) File passing is rather awkward, since it requires each main program to open, read, and parse every file. It also tends to clutter the environment with superfluous files; these can be deleted by commands at the end of a shell script, but must be removed by hand if the session is interactive or is aborted.

A more traditional subroutine-based programming style would allow communication by means of global variables and passed arguments as well as files. Display devices would also be under better control, since the device status can be remembered and maintained by the driver program, and each routine need not reallocate or reinitialize the display to be sure of getting a usable configuration.

In addition, the data files could be opened once, passed around, then closed

or deleted. Permanent files need be created only for permanent data, and the user need not specify the file names as arguments to every routine.

A final advantage of the subroutine approach is that there is less system overhead. The current approach requires that a UNIX (or EUNICE) process be created to run each command, since this is the only way to invoke a main program. The overhead in creating a process is much greater than that of calling a subroutine, and, in fact, may require several seconds of real time.

- *Operator Libraries*

As mentioned above, one advantage of the University of Maryland approach is the run-time compilation of customized image operators in order to reduce total execution time. The actual speedup achieved in the current RELAX package is rather small, but the technique could be extended for larger gains. Perhaps the greatest speedup could be achieved by using replicated in-line code instead of iterative constructs. For research purposes, of course, such optimizations are seldom worthwhile.

The run-time compilation of such routines is not a problem. It can be done as a separate program step, possibly invoked interactively by interrupting a session (with ^Y), compiling, and then resuming. It can also be done by using the "system" subroutine to call the compiler from within another program.

Since compilation is an expensive step, it might make sense to keep the most useful operators in a library of executable files. Nothing in the current RELAX package prevents this from being done, but neither are there any features to facilitate it. At the very least, a naming convention should be devised so that the operator names can be remembered.

- *Speedup*

The current relaxation updating algorithms are exceedingly slow. (They take about three CPU minutes for one pass of a 3×3 operator through a 128×128 image.) This is probably due to an inefficient scheme for accessing the floating-point label probabilities. Further investigation is needed to determine the requisite time for this type of processing.

Section 5

Program Documentation

This section constitutes a user's guide to the RELAX package as it is implemented on the SRI Image Understanding Testbed. As with any reference manual, it has occasionally been necessary to refer to terms before they are defined and discussed in detail. The first-time reader may find a preliminary scan through the section helpful. Additional information is available online, as described below.

5.1. Interactive Usage

There are typically five steps in applying relaxation to an image:

- Compilation of the updating routine
- Image-to-probability mapping
- Estimation of compatibility coefficients
- Relaxation updating iterations
- Probability-to-image mapping.

Display steps are usually interspersed so that one can watch the progress of the enhancement. Other techniques are sometimes required, such as edge detection or manual entry of neighborhood and compatibility data. All of these processes can be invoked from within the RELAX package.

The RELAX package currently takes no command-line arguments; just type "relax" and begin an interactive session. The following sample session displays some of the capabilities of the underlying CI driver language.

```
relax

  RELAX, Version 1.0

>
```

This invokes the program. You need not specify the full directory path name for the executable file if the path is given in your UNIX *.cshrc* shell startup file. (If you have no startup file, you may have to specify */iu/tb/bin/relax* or some other full path name.) The system then responds and waits for commands.

```
> *

      defcom        pcompat
      defnbr        prbimg
      erase         prelax
      hcompat       quit
      hrelax        setup
      imgprb        help
      insert
```

An "*" command lists all available commands. The *help* command is provided by the CI driver; all others are specifically related to the relaxation system. Typing "help" will give further information on the CI command interpreter and the help subsystem.

```
> help

Command names, variable names, and help topics
may be abbreviated to any unique prefix.  Several
instructions may appear on the same line, separated
by semicolons.  Use ^O to abort typeout and ^Z
to exit.

  command [ arg ... ]
    Execute a command with the specified arguments.

  foo*
    List commands that begin with "foo".
    Use just "*" to list all commands.

  foo* =
    List the names and values of variables
    that begin with "foo".

  variable [ param ... ]
    Display a variable value.  Some variables
    may require subscripts or parameters.

  variable [ param ... ] = value
    Assign a variable value.  The equals sign (=)
    may appear anywhere.

  ? topic  <or>  help topic  <or>  help *
    Print help message related to topic.  If topic
    is ambiguous, list the names of matching topics.

  push_level
    Creates a new level of the CI driver with the same
    commands available as there were at the preceding
    level.  The user may execute any combination of
    commands before quitting.

  ! command
    Fork a shell and execute the UNIX command.  If
    no command is given, just create an interactive
    shell process.

  < commandfile
    Read commands from "commandfile".

  : comment
    Accept a comment line; take no action.
```

As an example of the online help facility, we can print out the contents of the *setup.txt* file in the relaxation system help directory.

```
> help setup

setup [h|p] nlabels [ncols nrows]

  Setup is used to create programs for relaxation operations
  on images.  Both a compatibility operator and a relaxation
  operator will be compiled.  Either Hummel-Zucker-Rosenfeld
  (h) or Peleg (p) relaxation formulas may be chosen; the
  corresponding programs will be hcompat and hrelax or pcompat
  and prelax.  The default is "h".  You may also specify the
  number of class labels (default 2) to be used and the size
  of the relaxation neighborhood (default 3 x 3).
```

Running the *setup* command with default arguments produces a *param* file that is compiled into the *hrelax* and *hcompat* programs. The *param* file is then deleted and the executable programs are left in the current directory.

```
> setup

  Creating the Hummel-Zucker-Rosenfeld relaxation package ...

    Creating the "param" file ...
    Compiling hcompat ...
    Compiling hrelax ...
    Removing the param file ...
    Setup completed.
```

A UNIX directory listing command shows the new executable programs and a demonstration command file that was created previously.

```
> !ls

hcompat.exe     hrelax.exe      tank.cmd
  [continuing]
```

Here we use the command file to run a relaxation sequence. If the file were not in this directory, RELAX would search for it in directory */iu/tb/src/relax/demo*. RELAX then echos the commands and performs the indicated actions just as if they had been typed from the terminal.

```
> <tank.cmd
```

The image is first converted to two-label probability form. It has no pixel values below 13 or above 49, so stretching and clipping are requested. (This speeds convergence and improves the appearance of the output image.)

```
> : Convert the image to probability form.
> imgprb /iu/tb/pic/tank/bw.img 2 13 49
```

Next the compatibility coefficients are needed. They are computed from the pixel relationships in the original image. They could also be entered by hand with the *defnbr* and *defcom* commands. The program echos the *hcompat* request, with the default file names filled in.

```
> : Compute the compatibility coefficients.
> hcompat
  hcompat prb.img compat.dat
```

The original image is now to be reconstructed and displayed in an upper-left area of the screen. This reconstruction, which shows the input after stretching and clipping, also serves as a check on the *imgprb* and *prbimg* processes.

```
> : Display the (stretched) original image.
> erase
> prbimg output0.img
> insert output0.img 100 348
```

Now eight relaxation steps are to be run. Each will produce an output image; these images will be displayed along with the original in a 3 × 3 pattern.

```
> : Display eight relaxation steps.
> hrelax
  hrelax prb.img compat.dat

> prbimg output1.img
> insert output1.img 224 348

> hrelax
  hrelax prb.img compat.dat

> prbimg output2.img
> insert output2.img 348 348

> hrelax
  hrelax prb.img compat.dat

> prbimg output3.img
> insert output3.img 100 224

> hrelax
  hrelax prb.img compat.dat

> prbimg output4.img
> insert output4.img 224 224

> hrelax
  hrelax prb.img compat.dat

> prbimg output5.img
> insert output5.img 348 224

> hrelax
  hrelax prb.img compat.dat

> prbimg output6.img
> insert output6.img 100 100

> hrelax
  hrelax prb.img compat.dat

> prbimg output7.img
> insert output7.img 224 100

> hrelax
  hrelax prb.img compat.dat

> prbimg output8.img
> insert output8.img 348 100

End of command file.
```

The quit command is now given to return the user to the operating-system level.

```
> quit
```

5.2. Commands

The following commands are currently available within the RELAX system:

defcom compatfile relaxtype nlabels [neighborfile]
Defcom is an interactive program used to install manually computed compatibility coefficients for each neighbor of a point. The arguments specify the file to which the coefficients are to be written, whether a Hummel-Zucker-Rosenfeld ("h") or Peleg ("p") relaxation is desired, the number of labels in the relaxation process, and, optionally, a file that specifies a nonstandard neighborhood.

defnbr neighborfile ncols nrows
Defnbr is an interactive program used to define a nonstandard neighborhood for each point. The "neighborfile" argument specifies the file to which the neighborhood definition is to be written. The number of columns and rows that contain the neighborhood must also be specified. The program will ask which is to be considered the center point and whether each neighbor is to be considered as part of the neighborhood.

erase
Erase the display. Note that the display is not allocated or locked, so it will remain clear only if no one else transmits data to it.

hcompat prbfile compatfile [neighborfile]
Computes the Hummel-Zucker-Rosenfeld compatibility coefficients for the probability file (default *prb.img*) and stores them in a compatibility file (*compat.dat*). You may specify a neighborhood file as created by *defnbr*.

The *hcompat* program must have been compiled previously: see *setup*. If you would like to specify the compatibility coefficients by hand, see *defcom*.

hrelax prbfile compatfile [neighborfile]
Performs one Hummel-Zucker-Rosenfeld relaxation operation on a probability file (default *prb.img*), using compatibility coefficients in compatfile (*compat.dat*). You may specify a neighborhood file as created by *defnbr*.

The *hrelax* program must have been compiled previously: see *setup*.

imgprb inimg nlabels minval maxval
Convert an image to a probability format with the specified number of labels. The image will be clipped (stretched) using minval and maxval as the outer gray-level limits: omit these or specify -1 if the full input range is to be used. The file *prb.img* will be produced as output. At present it is a floating-point data file, not a true picture file.

insert picname mincol minrow
Insert the picture into the display at the specified lower-left pixel position.

pcompat prbfile compatfile [neighborfile]
Computes the Peleg compatibility coefficients for the probability file (default *prb.img*) and stores them in a compatibility file (*compat.dat*). You may specify a neighborhood file as created by *defnbr*.

The *pcompat* program must have been compiled previously: see *setup*. If you would like to specify the compatibility coefficients by hand, see *defcom*.

prbimg outname minval maxval
Convert a probability file to a luminance image format with the specified output range. Omit minval and maxval, or specify -1 to use the full (8-bit) dynamic range of the output image. The input file is assumed to be *prb.img*.

prelax prbfile compatfile [n neighborfile] [s nsets size1 ...]
Performs one Peleg relaxation operation on a probability file (default *prb.img*), using compatibility coefficients in compatfile (*compat.dat*). You may specify a neighborhood file as created by *defnbr;* precede it with the letter n.

You may also specify the grouping of label values into sets. Previously, in Section 3.2, we stated the Peleg updating rule as

$$p_i^{(t+1)}(\lambda_k) = \frac{1}{m}\sum_j \frac{p_i^{(t)}(\lambda_k) q_{ij}^{(t)}(\lambda_k)}{\sum_{\lambda=\lambda_1}^{\lambda_n} p_i^{(t)}(\lambda) q_{ij}^{(t)}(\lambda)} \qquad k=1,\cdots,n$$

where the denominator is the sum over the set of all labels. Actually, this sum can be limited to a set of labels, rather than summing over all labels. The sum is carried out over the set to which the label λ_k belongs.

$$p_i^{(t+1)}(\lambda_k) = \frac{1}{m}\sum_j \frac{p_i^{(t)}(\lambda_k) q_{ij}^{(t)}(\lambda_k)}{\sum_{\lambda\in[\lambda_k]} p_i^{(t)}(\lambda) q_{ij}^{(t)}(\lambda)} \qquad k=1,\cdots,n$$

Label sets are specified by the command option [s nsets size1 ...], where nsets specifies the number of different sets and size1, size2, ..., the number of labels in each set. The sets must consists of consecutive labels, *e.g.*, [s 3 2 4 3] specifies that there are 3 sets of labels; the first set contains the 2 labels 0, and 1; the second set labels 2, 3, 4, and 5; and the third set labels 6, 7, and 8.

Although the added flexibility of this label set grouping is available, its use is not recommended. The numbers computed by this method are no longer probabilities and the behavior of the process cannot be predicted[1].

The *prelax* program must have been compiled previously: see *setup*.

[1]Azriel Rosenfeld, personal communication, 1983.

quit n
Quit the current level of the CI driver. At the top level this will leave RELAX. The argument n may be provided to quit more than one level. Specify -1 or some large number to abort all levels of the driver and exit from the program.

setup {h|p} nlabels [ncols nrows]
Setup is used to create programs for relaxation operations on images. Both a compatibility operator and a relaxation operator will be compiled. Either Hummel-Zucker-Rosenfeld (h) or Peleg (p) relaxation formulas may be chosen; the corresponding programs will be *hcompat* and *hrelax* or *pcompat* and *prelax*. The default is "h". You may also specify the number of class labels (default 2) to be used and the size of the relaxation neighborhood (default 3 × 3).

5.3. Batch Execution

The RELAX program offers two methods of invoking prestored commands. The first is the interactive invocation of CI command files, as illustrated earlier. The second is to drive the entire RELAX session from an operating-system script. A UNIX shell script invoked by, *e.g.*, "runrelax picture.img 10 195", might look like the following:

```
# This is file "runrelax".
#
# Arguments $1 is the gray-level image.
# Arguments $2 and $3 are the optional low and
#   high clipping values.

relax <<!

  : Make 3 x 3 neighborhood, two label,
  : HZR coefficient computation and
  : relaxation programs.
  setup h 2

  : Display the original image, $1.
  erase
  insert $1 100 348

  : Transform the picture into a probabilistic image.
  imgprb $1 prb.img 2 $2 $3

  : Compute the compatibility coefficients.
  hcompat prb.img compat.dat

  : Apply the relaxation operator.
  hrelax prb.img compat.dat

  : Display the smoothed image.
  prbimg prb.img output1.img
  insert output1.img 224 348
!

echo "Finished."
```

Note that the shell substitution mechanism can be used. This is an advantage over interactive use, in which no substitution of variables is currently implemented.

To save the typed terminal output, you should pipe the standard output into a file. The UNIX method for doing this is to add *>session.log* to the *relax* command within the script or to the UNIX command line that invokes the script. You may also use the UNIX *script* or *tee* commands to route the typed output simultaneously to a file and to your terminal. (UNIX output buffering generally prevents you from interacting with a program while the script is being created. The EUNICE operating system does not have this limitation.)

The actual submission of this shell script is described in the UNIX programmer's manual. You should run it in foreground mode if you want to interact with the program. If you run it in background mode, be sure to pipe the output into a log file so that it won't appear on your terminal. You can monitor the log file during execution, using the *cat* or *tail -f* [formerly *tra*] commands to make sure everything is running smoothly, although the UNIX buffering mechanism prevents you from monitoring in real time. You can also halt the process or reconnect it to your terminal if you wish.

Section 6

Evaluation

This section documents the performance of the Hummel-Zucker-Rosenfeld and Peleg relaxation algorithms on a variety of imagery and in several scene analysis tasks. Although we have chosen realistic tasks, our goal was not the full exploration of relaxation techniques for these applications. Rather, we have chosen tasks with simple mapping functions to and from the probability domain so that we could better assess the actions of the probability-updating functions.

We could devise objective performance measures for particular tasks [Fekete81], but the relationship of such measures to subjective scene analysis performance would be difficult to quantify. Linear-feature extraction, for instance, must be rated differently when we desire only prominent features (*e.g.*, for stereo image matching) or closed boundaries of regions than when we desire extraction of all detectable discontinuities.

We could also compute performance measures for restoration of images to which we have added blur or noise, but we would need models of the imagery and degradations occurring in realistic scenarios. If such models were available, other methods of image restoration would almost certainly be more effective than heuristic relaxation. Furthermore, any comparison of relaxation output with ground truth would necessarily depend on the function used for mapping the initial image to the probability domain.

We have therefore chosen a subjective evaluation procedure. We compare the relaxation output with the probability input after each has been mapped back to the luminance domain. This type of comparison is valid for almost any task. It measures whether the relaxation operation has made the image more useful for the intended application.

The particular scene analysis tasks we have selected are discussed below. For any task and type of imagery, we still must choose the

- Image-to-probability and inverse mappings
- Size (and shape) of the compatibility neighborhood
- Method of computing compatibility coefficients
- Relaxation scheme (HZR or Peleg)
- Number of relaxation iterations.

We shall explore these choices in the following subsections and then summarize our conclusions.

6.1. Task Selection

The RELAX package supplied by the University of Maryland contained a demonstration of noise suppression on an infrared image of a military tank. The nature of the image, a single bright tank centered against a dark background, made this also a demonstration of object detection and of segmentation in a noisy image. We have

used this image (and the implied tasks) extensively in learning to run the RELAX package and to understand its functioning. Two other test tasks we have selected are edge linking and anomaly detection.

Edge linking is the enhancement of linear features in an image by strengthening connected edge elements and suppressing isolated or conflicting ones. A set of edge operators is first passed over the image; they return, for each pixel, the probabilities that scene edges at various orientations are present. (We shall treat *no-edge* as a label also.) The relaxation stage strengthens the probabilities of edge labels that link "nose to tail" with the edges at neighboring pixels while suppressing other edge pair orientations.

Anomaly detection is the identification or enhancement of isolated blobs against a fairly uniform background. An image operator first classifies pixels as either background or anomalous according to the statistics of the background and the gray level of each pixel. Relaxation then reinforces the classification of a pixel if its neighboring pixels are similarly classified. This two-label operation may also be viewed as a noise-suppression application.

6.2. Edge Linking

The first step in edge linking is to calculate the initial probabilities that scene edges at various orientations pass through each pixel. We convolve the image with four Sobel-type masks that detect *northwest*, *north*, *northeast*, and *east* edges. Because we consider opposing gradient directions (or contrasts) to be equivalent, there are four orientations rather than eight.

The 3 × 3 convolution operators are:

0	2	1	1	0	−1	1	2	0	1	2	1
−2	0	2	2	0	−2	2	0	−2	0	0	0
−1	−2	0	1	0	−1	0	−2	−1	−1	−2	−1
northwest			***north***			***northeast***			***east***		

Five labels are then attached to each pixel: *no-edge*, *northwest*, *north*, *northeast*, and *east*. The probabilities for each of these labels are calculated from the outputs of the four convolution operators, $Op(dir)$. For label *north*, the probability at a particular pixel is calculated as

$$Prob(north) = \frac{Op(north)}{\sum_{all\ dir} Op(dir)} \cdot \frac{\max_{all\ dir} Op(dir)}{\max_{\substack{all\ dir,\\ all\ image\\ pixels}} Op(dir)},$$

and similarly for $Prob(northwest)$, $Prob(northeast)$, and $Prob(east)$. The first term relates the $Op(north)$ response to the response of the other edge operators; the second compares the operator response at that pixel with responses over the entire image.

The probability of *no-edge* is calculated as

Evaluation

$$Prob(no\text{-}edge) = 1 - \frac{\max_{all\ dir} Op(dir)}{\max_{\substack{all\ dir,\\ all\ image\\ pixels}} Op(dir)}.$$

Together these probabilities constitute the initial probability image. Similar schemes have been reported in the literature [Riseman77, Zucker77].

The next step is to calculate the compatibility coefficients. We have generally used the RELAX package *hcompat* and *pcompat* routines for that purpose. Our aim was to evaluate fully automated relaxation rather than to develop optimal compatibility matrices for this one application; the former is the usual approach in the relaxation literature.

Figures 1-13 illustrate the results of our edge-linking efforts. Each figure consists of sixteen pictures. The upper-left picture is the original image. The other three pictures across the top of the figure and the leftmost picture on the second row are the *northwest*, *north*, *northeast*, and *east* edge maps. In these images, black represents zero probability of an edge, white a unity probability.

The remaining pictures are binary images. The second from the left in the second row is the inverse mapping of the original probability image. It is produced from the four edge maps by displaying, for each pixel, the label with the highest probability; *no-edge* maps to black while all other labels map to white. The remaining ten pictures are obtained by successive iterations of the relaxation procedure. They represent the results of 1, 2, ..., 10 iterations of relaxation. Figure 2 is the exception: the ten pictures are the results of 10, 20, ..., 100 iterations.

Figures 1 and 2 illustrate a fairly typical relaxation application. The first few iterations of relaxation show a strong edge-linking effect. Later iterations seem to do little except smooth or blur the enhanced structures. This dual nature of relaxation has been analyzed by Richards [Richards80]. Figure 3 shows that the Peleg relaxation scheme exhibits essentially the same behavior.

Figure 4 illustrates the HZR method on aerial imagery. The improvement in the displayed output is rather dramatic, although it may be partially due to the method of output mapping.

Figures 5 and 6 illustrate the effect of larger neighborhoods. The 7 × 7 neighborhood increases edge spreading for the HZR method and decreases it for the Peleg method. Computation time is much greater for large neighborhoods, of course.

Figure 7 shows that strong edge structure does not guarantee effective compatibility coefficients if the edges appear equally in many orientations and relationships. This is opposite to the effect of most enhancement operators.

Figure 8 shows the result of "enhancing" the edges in an image that has no perceptual edge structure. The procedure for computing compatibility coefficients registers any regularities in the image, not just strong responses from the edge detectors. This, combined with the relaxation blurring effect, gives a surprisingly good noise-suppression or object detection operator.

Figures 9 through 12 show that the exact values of the compatibility coefficients are not critical. Figure 9 was made with coefficients rounded to one decimal place, Figure 10 with these same numbers doubled. (Doubling the coefficients will have no effect on Peleg relaxation.) Figure 11 was generated using compatibility coefficients

derived from the image in Figure 12; Figure 12 was likewise generated using compatibility coefficients derived from the image in Figure 11. In each case the tampering had relatively little effect.

Figure 13 shows what happened when we supplied the coefficients by hand to see if a careful choice of the values could produce faster relaxation effects. Although these coefficients were intuitively derived, they perform very badly. For a pixel and the neighbor above it, the coefficients we used for the Hummel-Zucker-Rosenfeld scheme are

$$
\begin{aligned}
Coeff(north, northwest) &= 0.50 \\
Coeff(north, north) &= 1.00 \\
Coeff(north, northeast) &= 0.50 \\
Coeff(north, east) &= -0.50 \\
Coeff(north, no\text{-}edge) &= -0.25 \\
Coeff(northeast, northwest) &= -0.50 \\
Coeff(northeast, northeast) &= -0.75 \\
Coeff(northeast, east) &= -0.50 \\
Coeff(northeast, no\text{-}edge) &= 0.50 \\
Coeff(east, east) &= -0.75 \\
Coeff(east, no\text{-}edge) &= 0.75
\end{aligned}
$$

For a pixel and its northeast neighbor we used

$$
\begin{aligned}
Coeff(north, northwest) &= -0.50 \\
Coeff(north, north) &= 0.25 \\
Coeff(north, northeast) &= 0.25 \\
Coeff(north, east) &= -0.25 \\
Coeff(north, no\text{-}edge) &= 0.50 \\
Coeff(northeast, northwest) &= -0.5
\end{aligned}
$$

(The other required compatibility coefficients can be derived from these by using symmetry considerations.) We also tried variations of the above, *e.g.*, reversing the signs of some coefficients; there were no significant changes in results. The lesson seems to be that manual selection of coefficients is exceedingly difficult. A better policy is to generate coefficients using *hcompat* or *pcompat* and to modify the values only slightly.

6.3. Anomaly Detection

Anomaly detection is a two-label problem: each pixel is part either of the background or of an anomaly. We assume that the background comprises the majority of the image and that it can be modeled as a constant gray level plus Gaussian noise. Anomalies are regions that diverge significantly from this model. The relaxation process should strengthen the probability of an *anomaly* label for dark or light groups of pixels while suppressing the *anomaly* label for isolated pixels that are equally bright.

The background is modeled by the mean gray level and standard deviation for the whole image. Then, for each pixel in turn, we calculate the deviation of the pixel value from the background value (*i.e.*, from the image mean). The probability that

the pixel is a background pixel is then taken to be the probability that a deviation this large or larger can occur by chance. The probability that the pixel is part of an anomaly is taken to be 1.0 minus this value.

It is somewhat easier to present the mathematical formulas in the reverse order. Let us suppose that a pixel is t standard deviations from the image mean. We calculate the probability of the label *anomaly* as

$$Prob(anomaly) = \int_{-t}^{t} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx,$$

$$Prob(background) = 1 - Prob(anomaly).$$

These initial probabilities are used to calculate the compatibility coefficients by means of the *hcompat* or *pcompat* routine. They also serve as input to the relaxation process. After a number of relaxation iterations, the pixels for which the *anomaly* label has the higher probability are displayed as white points in a binary image.

Figures 14-16 are the results of our efforts at anomaly detection. Each figure is a pair of images that document the results for one example. The four pictures in the top image, from left to right, are the

- Original scene.
- Anomaly probabilities mapped to gray levels 0 to 255.
- Binary image formed by inverse mapping the two-label probability image with the *prbimg* routine.
- Binary result of one iteration of relaxation.

The bottom image shows, from left to right, the results of 2 through 5 iterations of relaxation.

Figure 14 begins with the original gray-level image of a composite road scene. Figure 15 is similar, but derived from a version of the road scene that has been "cleaned" to remove the background road profile and make the vehicles (anomalies) more distinct. (This technique is part of the SRI road tracker [Quam78].) Figure 16 is the Peleg method applied to the cleaned road scene. In each case, relaxation enhances the background noise structure until it swamps the anomaly signal.

The lesson from this sequence is that a good initial image does not guarantee a good result. The compatibility coefficients derived by *hcompat* or *pcompat* do not necessarily encode our own notions of image "structure." Relaxation-based anomaly detection may be feasible, but not by using the straight-forward approach attempted here.

6.4. Summary

For any task and type of imagery, the user must choose the

- Image-to-probability and inverse mappings
- Size (and shape) of the compatibility neighborhood
- Method of computing compatibility coefficients

- Relaxation scheme, HZR or Peleg
- Number of relaxation iterations.

We can now give some guidance on these matters as well as on the question of whether to use relaxation at all.

We have presented some simple image-to-probability and inverse mappings. Each mapping is designed for a specific application, and the success of relaxation processing depends critically on the quality of the initial mapping. While we can say little about the initial imagery or the mapping function, we can suggest some constraints on their combination—the probability image.

The probability image is the source of both the compatibility coefficients and the relaxation output. Any repetitive structure in this image will be reflected in the coefficients and enhanced in the output image. It is therefore essential that the probability image have the following characteristics:

- The signal, or desired characteristics, should dominate the noise.
- The signal must be spatially correlated within the chosen neighborhood; the noise, however, should be uncorrelated.
- The signal must contain some spatial relationships and not others; the noise should contain all relationships equally.

Relaxation will be successful to the extent that these conditions are met.

The user should specify a neighborhood just large enough to guarantee the above conditions. An application with large contiguous regions, such as land use classification, might benefit from the noise immunity of a large neighborhood. Generally, though, a 3×3 neighborhood is the best choice. If the signal does not dominate the noise locally, it is unlikely to dominate it within larger neighborhoods. If it does dominate on the average, relaxation provides a mechanism for propagating the signal a short distance into those regions where the signal is weak. Larger neighborhoods would increase penetration distance, but at the cost of greatly increasing the computation time.

Different compatibility coefficients are needed by different relaxation methods, but the user is faced with similar choices in computing them. They may be computed separately for each image, jointly for an ensemble of images, or theoretically for some image model. The same set of constraints applies: the sampled or modeled population must have biased second-order statistics rather than an equiprobable selection of spatial relationships. This typically prohibits training on an ensemble of randomly oriented images.

The RELAX *hcompat* and *pcompat* routines look for combinations of neighboring labels that occur more (or less) frequently than that expected if the label assignments were statistically independent. These estimates are calculated under the assumption that the image is a homogeneous data set, ignoring the fact that the structure in one part of the image may be significantly different from that of other parts. It has been suggested that this disadvantage may be alleviated by carrying out the calculation over windows of the image [Nagin82]. However, the problem of estimating coefficients from small samples worsens as the sample size decreases. If we have many labels, a large neighborhood, and a small sampling area, the number of samples with similar pixel configurations will be small and the estimated compatibility coefficients will be unreliable.

An alternative method for obtaining compatibility coefficients is for the user to

estimate them. For many labels and a large neighborhood, manual entry of the coefficients is a daunting task. Moreover, except for obvious cases of compatibility, it is difficult to arrive at good coefficient values by guessing. Figure 13 shows the hopeless results we obtained when we guessed what we believed to be reasonable values for the coefficients. A major difficulty was our reluctance to allow independent label combinations. Generally we assigned values that forced combinations to be either compatible or incompatible rather than independent. If manually entered values of the coefficients are to be used, we suggest considering those the package can provide as a guide to assigning reasonable values; slight modifications may then be beneficial.

Which relaxation scheme should be used? It does not seem to matter. One method sometimes does a little better than the other, but both work or fail together. The principal distinction we observed was that the relaxation smoothing effect could be reduced for the Peleg method by using a larger neighborhood.

The last question that needs to be answered is how many iterations of relaxation should be performed. There appear to be two aspects to the relaxation process. The first three or four iterations often show moderate enhancement, while later ones are often dominated by blurring. Little change occurs after ten iterations. Only by looking at the output can one ascertain the optimum halting point, but approximately four iterations seems to be a good rule of thumb.

Relaxation is essentially an enhancement operator, with the structure to be enhanced derived from the image rather than from any model of either the imagery or the application. What is the cost of this signal enhancement? One iteration of relaxation, using a 3×3 neighborhood over a 128×128 image, took approximately three CPU minutes on a VAX 11/780—a considerable cost if image smoothing is the desired result. These costs increase linearly with the number of image pixels, the number of pixels in the neighborhood, and the number of iterations.

Evaluation

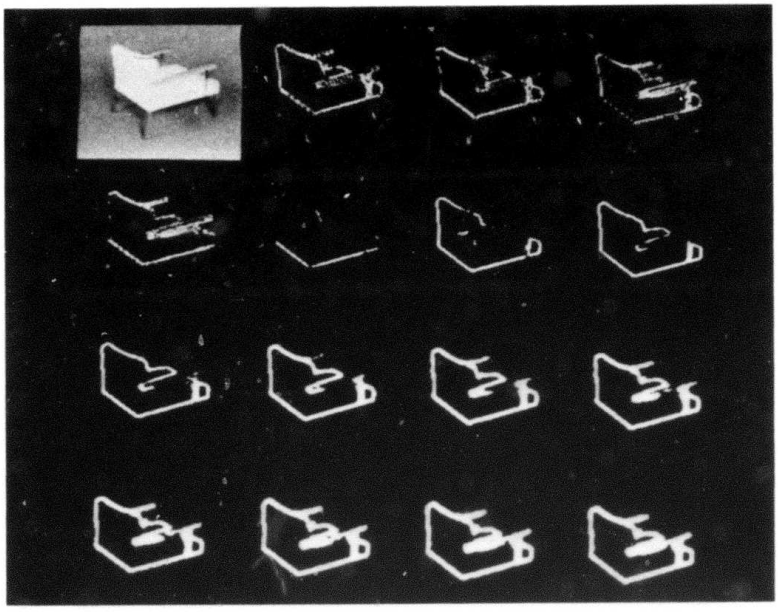


Figure 1. HZR relaxation using a 3×3 neighborhood and *hcompat* compatibility coefficients. See text for explanation of the figure format. Edge linking during the first iterations is supplanted by blurring or spreading.

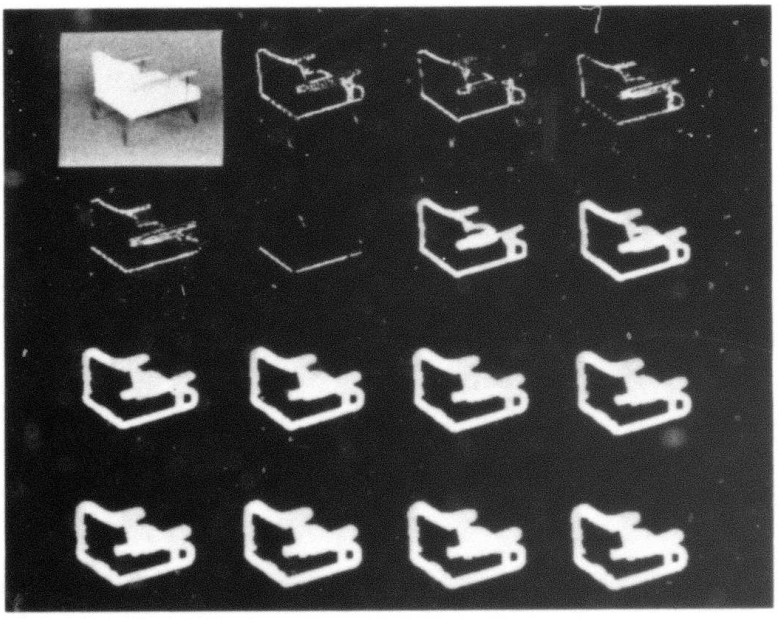


Figure 2. Further iterations of the example shown in Figure 1. Iterations 10, 20, ..., 100 are shown. These additional iterations only spread the edges.

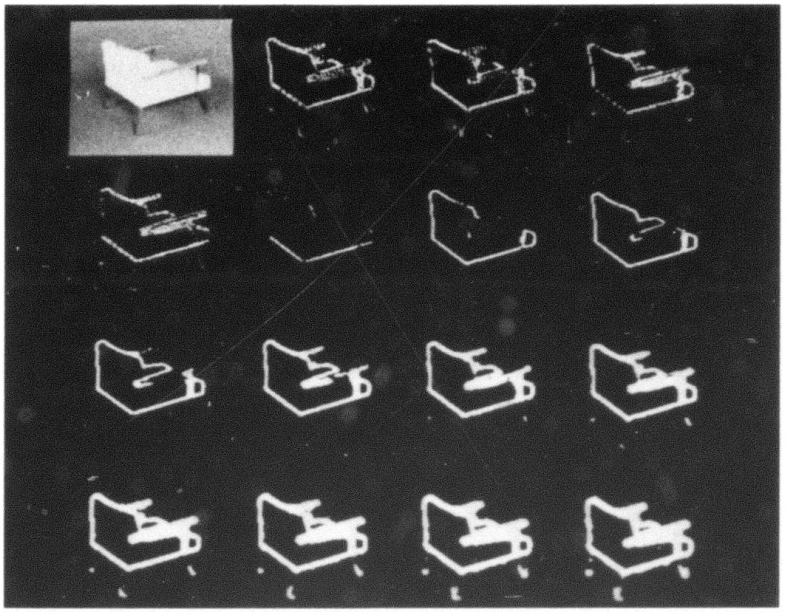


Figure 3. Peleg relaxation using a 3 × 3 neighborhood and *pcompat* compatibility coefficients. Results are similar to those obtained using the HZR scheme. See Figure 1 for comparison.

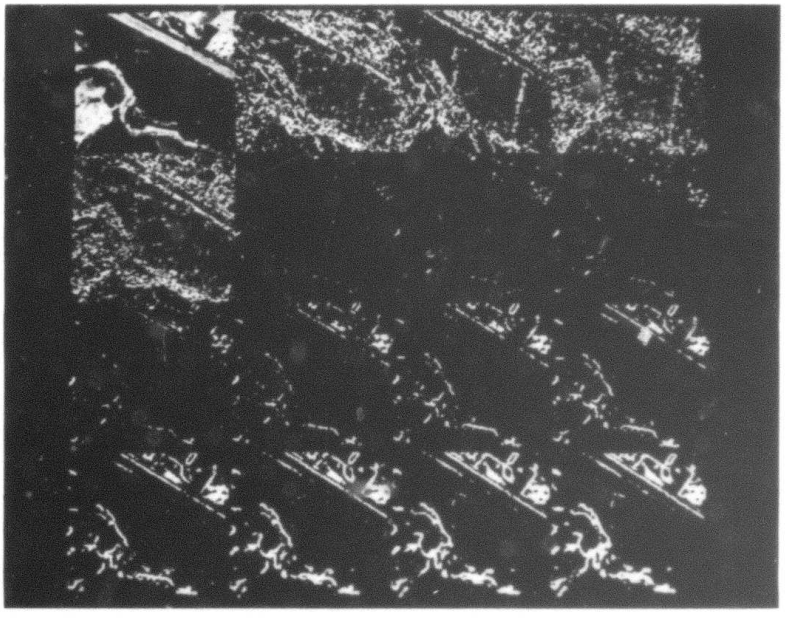


Figure 4. An aerial image with HZR relaxation using a 3 × 3 neighborhood and *hcompat* compatibility coefficients. Relaxation extracts structure from noisy edge data.

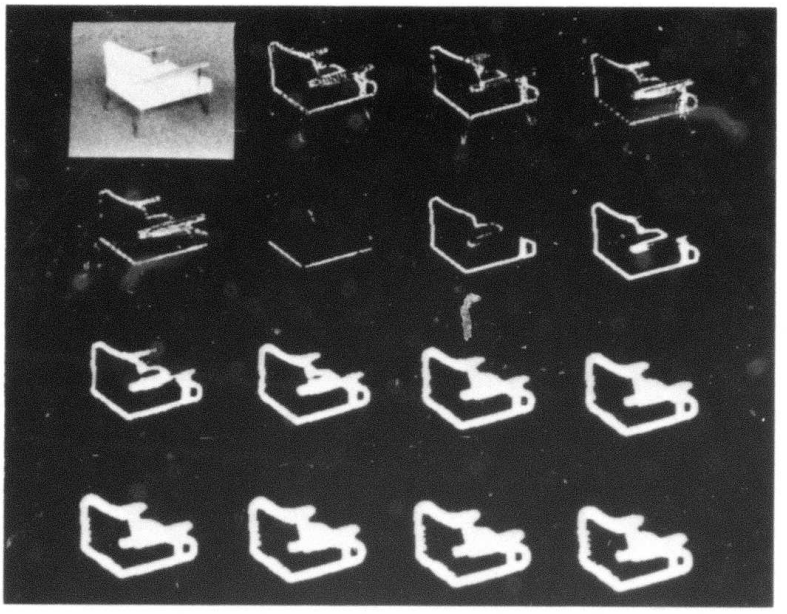


Figure 5. HZR relaxation using a 7 × 7 neighborhood and *hcompat* compatibility coefficients. The use of a larger neighborhood in the HZR relaxation scheme generally increases edge spreading.

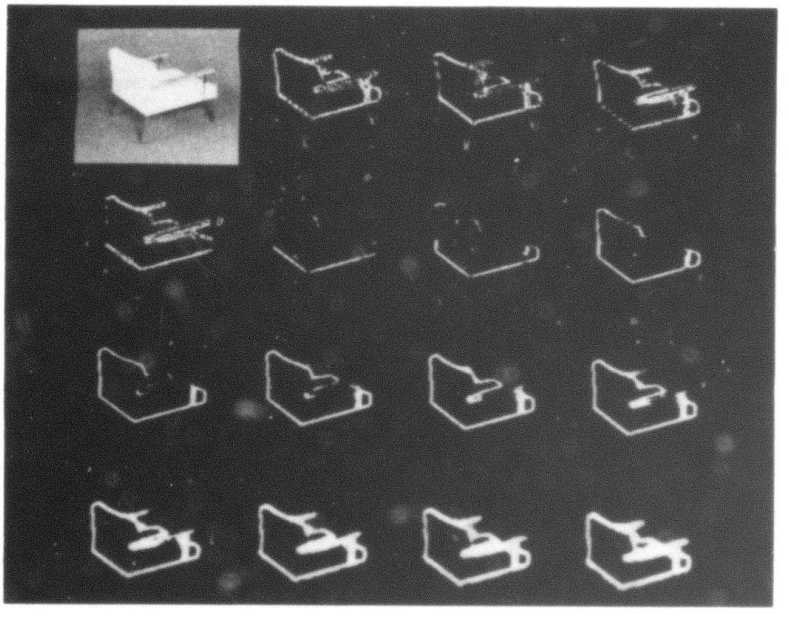


Figure 6. Peleg relaxation using a 7 × 7 neighborhood and *pcompat* compatibility coefficients. The larger neighborhood reduces the edge spread instead of increasing it.

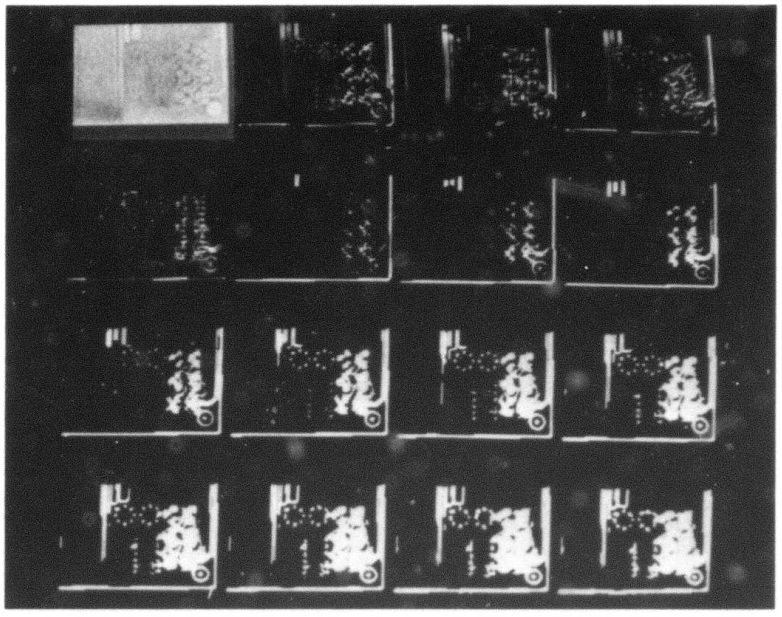


Figure 7. A printed-circuit board image with HZR relaxation using a 3 × 3 neighborhood and *hcompat* compatibility coefficients. Both this example and Figure 1 have sharp edges, but this example lacks bias in the edge orientations.

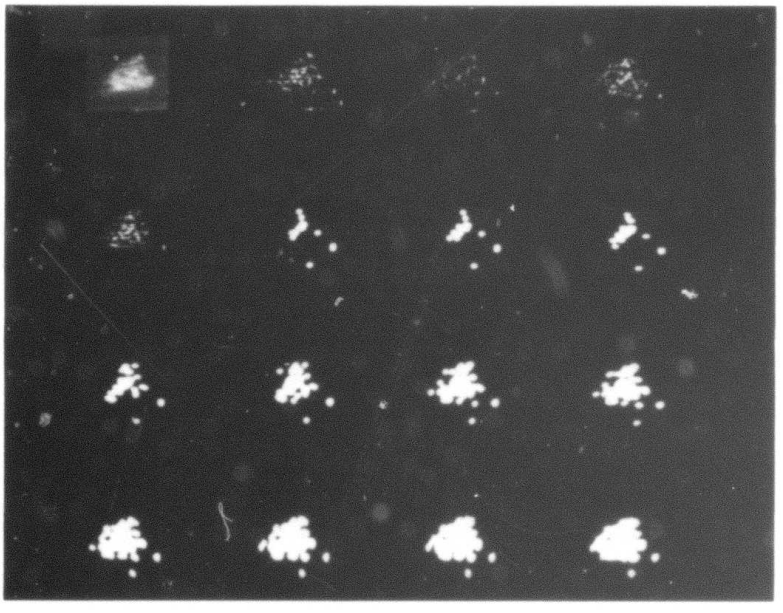


Figure 8. A noisy tank image with HZR relaxation using a 3 × 3 neighborhood and *hcompat* compatibility coefficients. Although the edge-detection method is inappropriate here, it works quite well for smoothing or object detection.

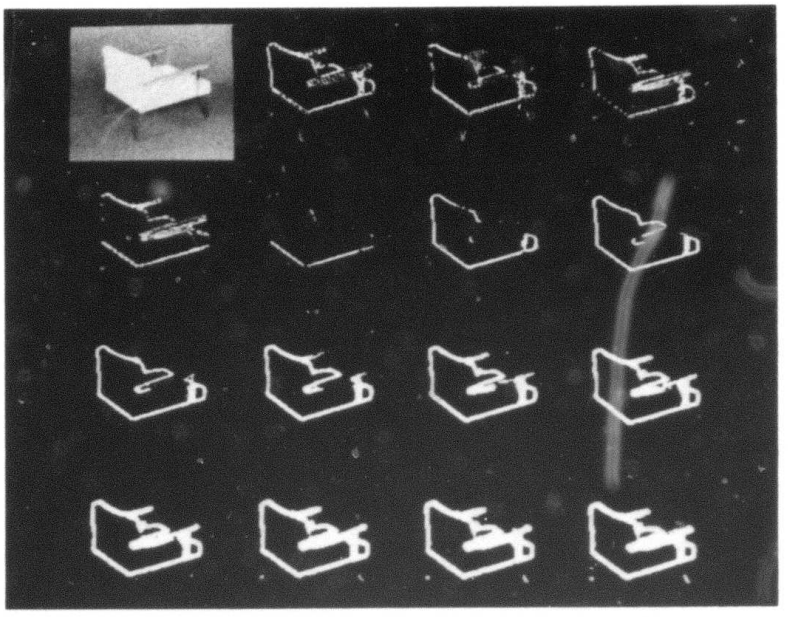


Figure 9. HZR relaxation using a 3 × 3 neighborhood and *hcompat* compatibility coefficients rounded to one decimal place. Comparison with Figure 1 shows that exact values of the coefficients are not important.

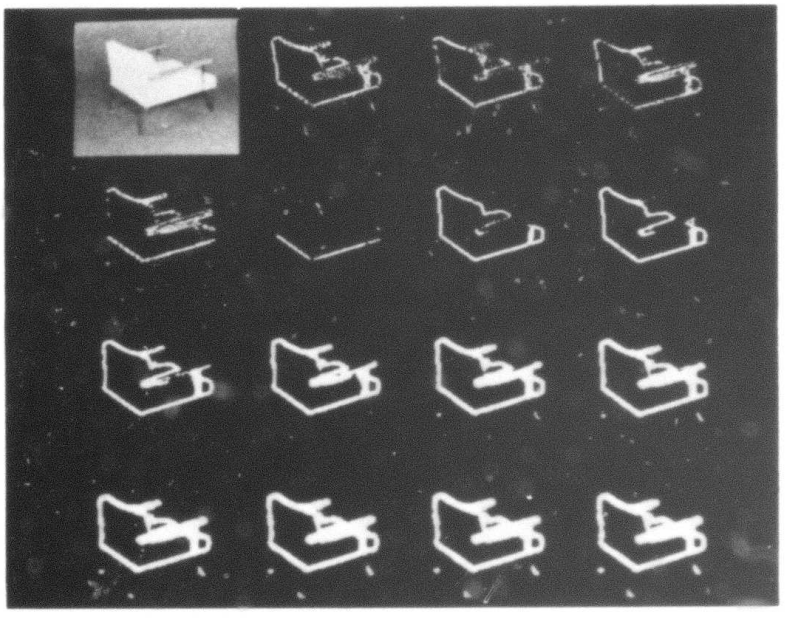


Figure 10. HZR relaxation using a 3 × 3 neighborhood and *hcompat* compatibility coefficients that are twice those for Figure 9. Only the relative values of the compatibility coefficients are important.

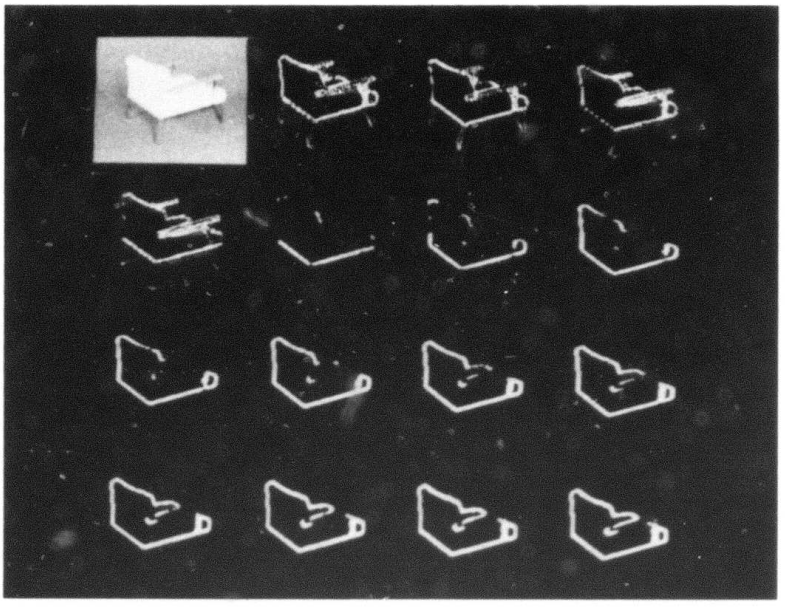


Figure 11. HZR relaxation using a 3 × 3 HZR and compatibility coefficients derived from the aerial scene in Figure 12. Useful compatibilities may be derived from dissimilar imagery.

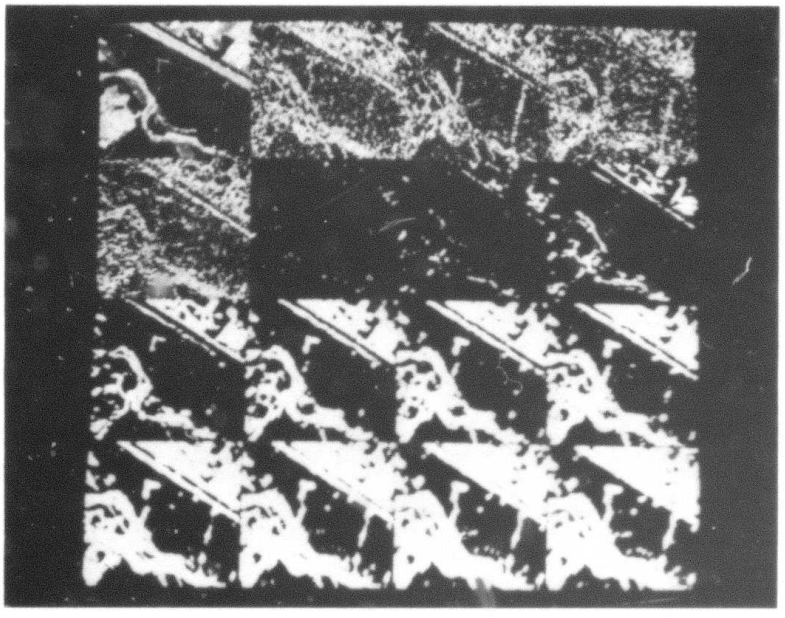


Figure 12. HZR relaxation using a 3 × 3 neighborhood and compatibility coefficients derived from the chair scene in Figure 11. Useful compatibilities may be derived from dissimilar imagery.

Evaluation

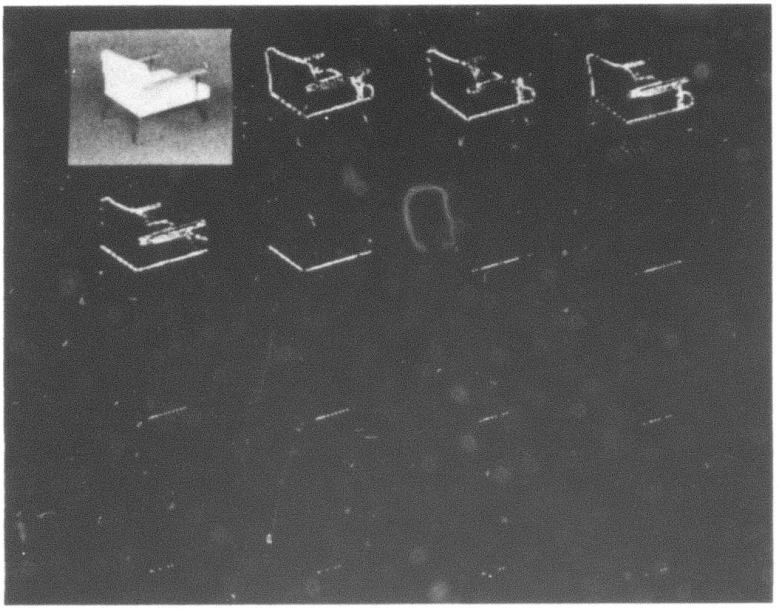


Figure 13. HZR relaxation using a 3 × 3 neighborhood and manually selected compatibility coefficients; see text for details. Although these coefficients were intuitively determined, they perform very badly.

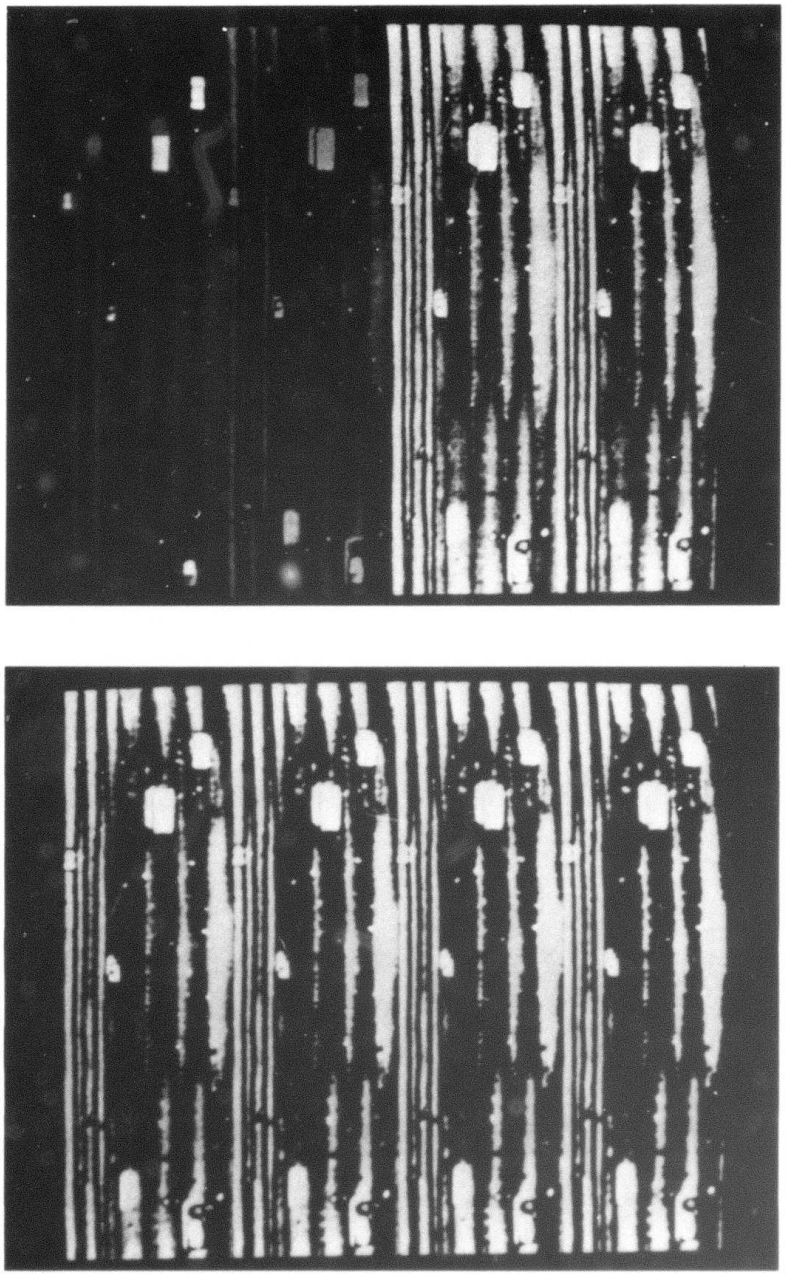


Figure 14. Anomaly detection by HZR relaxation using a 3 × 3 neighborhood and ***hcompat*** compatibility coefficients. Top image: the original scene, the anomaly probabilities, and binary outputs of iterations 0 and 1. Bottom image: iterations 2 through 5. Spatially correlated noise is enhanced along with the signal.

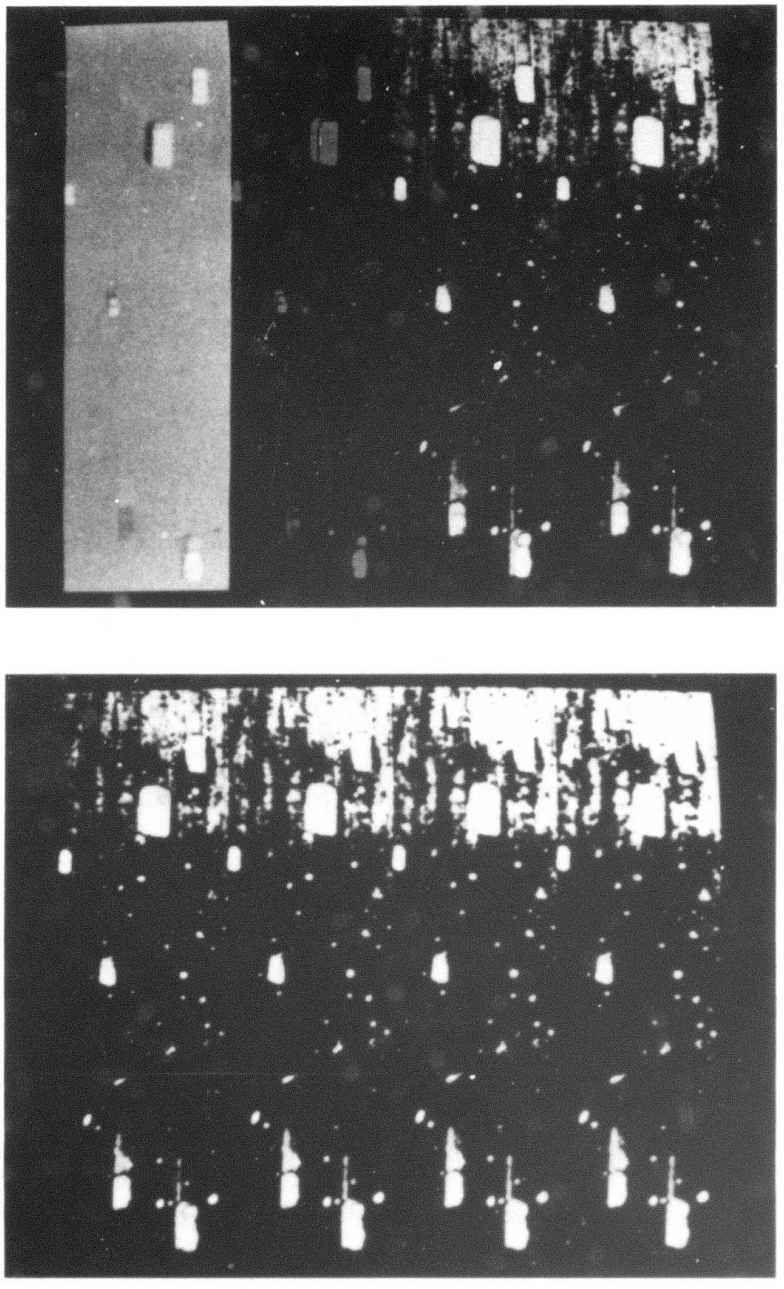


Figure 15. Anomaly detection by HZR relaxation using a 3 × 3 neighborhood and ***hcompat*** compatibility coefficients. These results are obtained from a "cleaned" version of the original scene used in Figure 14.

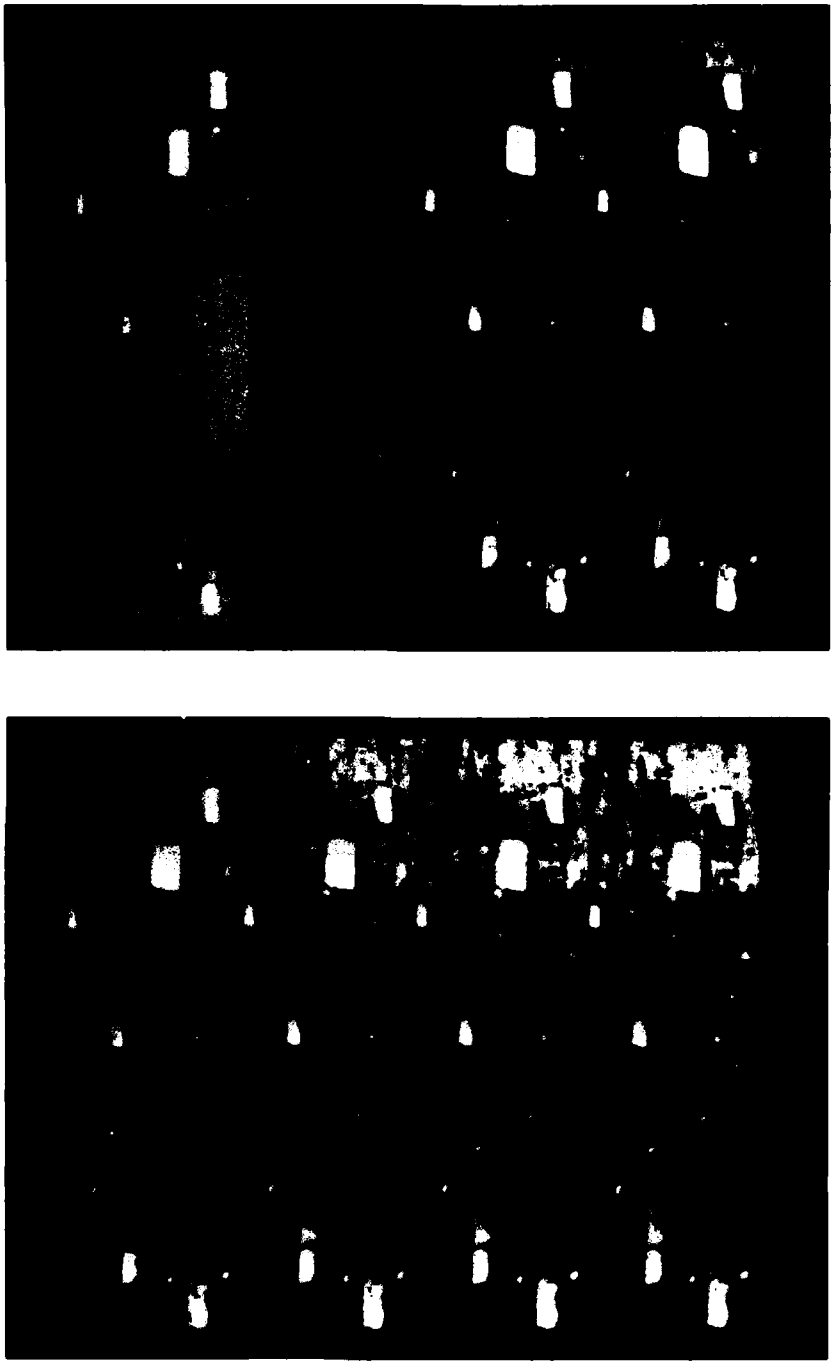


Figure 16. Anomaly detection by Peleg relaxation using a 3 × 3 neighborhood and *pcompat* compatibility coefficients. Comparison with Figure 15 shows that HZR and Peleg methods produce similar results.

Section 7

Suggested Improvements

The process of evaluation has turned up several ways to improve or extend the current RELAX implementation. Comments about existing features have been made at the appropriate junctures throughout this document. The following are additional suggestions for substantial modifications or needed research.

- ***Improved Coefficient Entry and Editing***

  Manual entry of compatibility coefficients is currently very awkward, and once entered the coefficients cannot be displayed or altered. The Testbed *view* program was developed to display files of coefficients, but a more flexible display-and-editing capability is needed within the relaxation package itself.

  One way to reduce the burden on the user is to use symmetry or other constraints to reduce the number of coefficients that must be typed in. Another is to allow entry of important individual coefficients, with all others defaulting to the central value (0.0 or 1.0). (This approach could be extended to the relaxation updating formula, with only important terms actually being entered into the computation.) In any case, a coefficient query and correction capability would be very useful.

- ***Ensemble Coefficient Extraction***

  The current *hcompat* and *pcompat* routines extract compatibility coefficients from one probability image. There may be applications for which average coefficients from an ensemble of similar images are desired. A simple program could be written to extract coefficients from such an ensemble, or to combine coefficient matrices derived from individual images.

- ***Supervised Learning***

  Relaxation enhancement is often unpredictable when the compatibility coefficients are derived from noisy images. One could argue that "cleaned" or "ground truth" images should be used for deriving the coefficients. This would build signal statistics into the compatibility coefficients directly instead of depending on desired signals to dominate the noise in the initial probability image. The result should be faster convergence and better signal enhancement [Peleg78a, Eklundh80].

Suggested Improvements

- *Adaptive Coefficients*

  Since relaxation can be used to enhance signals and suppress noise, it may be useful for producing cleaned images for the estimation of compatibility coefficients. In the limit, new coefficients could be extracted during (or after) each application of relaxation updating. This would either produce faster enhancement or faster degradation of the image.

- *Use of Decision Logic*

  The Hummel-Zucker-Rosenfeld or Peleg updating formulas in the RELAX package may be well suited for enhancement and smoothing applications. Many other uses of iterative image modification would require nonlinear decision logic in the updating algorithms. The decision logic might make use of the image histogram, the statistics of objects already found in the image, or other global information.

- *Use of Joint Neighborhood Constraints*

  The current probability updating formulas use only pairwise relationships to compute a new pixel label probability. Other types of iterative image operators often look for patterns in the neighborhood as a whole. There may be relaxation applications for which joint neighborhood relationships must similarly be modeled. See Peleg [Peleg80a] for a discussion of the conditional independence assumption.

- *Adaptive Neighborhood Definition*

  A particular use of joint neighborhood constraints and decision logic is to decide, for each neighborhood, which pixels belong to the same region as the central pixel. Only those pixels would be used in updating the central-pixel label probabilities. This should speed convergence in segmentation applications.

- *Halting Criteria*

  The RELAX package currently offers no way to ascertain how many iterations of relaxation updating are sufficient for any given task. We have suggested that three or four iterations are usually optimum for enhancement applications, but there are no image-dependent rules for determining when improvement has stopped and blurring has taken over. More research is necessary in this area. See Fekete *et al.* [Fekete81] for an approach based on examining the rates of change and the entropies of the probability vectors at each iteration.

Suggested Improvements

- *Further Research*

  We have attempted to evaluate relaxation as a technique rather than make an exhaustive study of its application to a particular task. If relaxation seems promising for a specific task, however, such a thorough evaluation may be required. As relaxation techniques are still in their infancy, further research is needed to determine where and how they may be best applied.

Section 8

Conclusions

Relaxation is a procedure for enhancing the signal, or features, found in an image by an imperfect enhancement, detection, or classification operator. It is a very general technique and has been used in a variety of image-processing applications.

The approach works when a label at one image pixel is constrained by neighboring labels. The relaxation procedure discovers and exploits these relationships to produce a more consistent labeling. Where an initial label is strongly believed, it tends to be unchanged by relaxation updating. Where it is uncertain, relaxation tends to propagate either the neighborhood information or its own biases into the classification. This results in either enhancement or smoothing, depending on the nature of the compatibility coefficients.

There are three basic components to relaxation: mapping the original image to a probability domain, estimating the compatibility coefficients, and applying the updating formula. The updating formulas in the RELAX package are simple, standard, and nearly equivalent, so that only the first two components are of concern.

Each application domain requires a different mapping to the probability image format. The mapping should be such that (1) the desired signal dominates other image components; (2) the signal is locally correlated and occurs in only certain neighboring combinations; (3) the noise is locally uncorrelated and appears in all possible combinations.

Compatibility coefficients can be provided by hand, although they are exceedingly difficult to derive for most applications. The automated coefficient extraction routines in the RELAX package work well if the mapping constraints above are met, but produce surprising results otherwise. If the noise or unwanted signal in an image is spatially correlated, it will be enhanced. If the desired signal takes on all possible local relationships, it will not be enhanced. Enhancement using image-based compatibility coefficients can improve on a good initial image, but will not redeem an incompetent detection operator.

Relaxation methods for solving "gravitational" or "fluid flow" problems such as histogram sharpening, requantization, image smoothing, and classification-map improvement have been reported in the literature. Relaxation can be used for model-independent enhancement, but is often more costly and less effective than model-based enhancement or restoration when appropriate models are available.

The RELAX package provides a mechanism for exploring the relaxation philosophy in image-based applications. Relaxation techniques are still in an early stage of development, and more research is needed into both theoretical foundations and domains of applicability.

Appendix A

The GPSPAR Relaxation Package

This appendix documents the control language used by the original University of Maryland contribution to the Testbed, the GPSPAR relaxation package. This is a set of stand-alone programs that may be invoked in sequence, either interactively or using a UNIX shell script.

The capabilities and user interfaces of the GPSPAR programs are essentially identical to those documented for the interactive RELAX driver. (We have changed the names of some of the programs during integration into the Testbed; for instance *prbimg* was originally known as *display*. The shell script is just another method of invoking these programs. A sample script is shown below.

```
# This program will do everything a person sitting
# at a terminal would do to:
#
#   Set up compatibility coefficient creation and
#   relaxation programs using the Hummel-Zucker-Rosenfeld
#   formulas,
#
#   Create a two label probabilistic image from the
#   gray level "tank" picture (or other image) using
#   the problem-specific program "imgprb",
#
#   Compute the compatibility coefficients from this
#   image using the program "hcompat" that was produced
#   by "setup",
#
#   Perform eight iterations of relaxation using "hrelax"
#   as well as converting each resulting probabilistic
#   image into a gray level image, output.img, using the
#   "prbimg" program.


# Erase the screen.
erase


# Make 3 x 3 neighborhood, two label, HZR coefficient
# computation and relaxation programs.

csh /iu/tb/bin/setup.csh h 2


# Transform the picture into a probabilistic image.
# $1 is the image name, $2 and $3 are the optional
# low and high pixel range specifications.

if ($#argv < 1) then
  imgprb /iu/tb/pic/tank/bw.img prb.img 2 13 49
else
  imgprb $1 prb.img 2 $2 $3
endif
```

```
# Recreate a gray level image from this.  This first
# "output" image will have gone through essentially
# the same steps as the other new gray level images
# to be created.  The output is stretched to fill
# 8-bit pixels.

prbimg prb.img output0.img
show output0.img -t 100 348


# Compute the compatibility coefficients from the
# probabilistic image.
hcompat prb.img compat.dat


# Perform eight iterations of relaxation on the image.
# After each iteration display a gray level image
# representation.

hrelax prb.img compat.dat
prbimg prb.img output1.img
show output1.img -i -t 224 348

hrelax prb.img compat.dat
prbimg prb.img output2.img
show output2.img -i -t 348 348

hrelax prb.img compat.dat
prbimg prb.img output3.img
show output3.img -i -t 100 224

hrelax prb.img compat.dat
prbimg prb.img output4.img
show output4.img -i -t 224 224

hrelax prb.img compat.dat
prbimg prb.img output5.img
show output5.img -i -t 348 224

hrelax prb.img compat.dat
prbimg prb.img output6.img
show output6.img -i -t 100 100

hrelax prb.img compat.dat
prbimg prb.img output7.img
show output7.img -i -t 224 100

hrelax prb.img compat.dat
prbimg prb.img output8.img
show output8.img -i -t 348 100

#   done
echo "Finished."
```

References

[Barrow76] H.G. Barrow and J.M. Tenenbaum, *MSYS: A System for Reasoning about Scenes*, Technical Note 121, Artificial Intelligence Center, Stanford Research Institute, Menlo Park, California, April 1976.

[Bhanu82] B. Bhanu and O.D. Faugeras, "Segmentation of Images Having Unimodal Distributions," *Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 4, pp. 408-419, July 1982.

[Brice70] C.R. Brice and C.L. Fennema, "Scene Analysis Using Regions," *Artificial Intelligence*, Vol. 1, No. 3, pp. 205-226, Fall 1970.

[Davis77a] L.S. Davis and A. Rosenfeld, *Noise Cleaning by Iterated Local Averaging*, Technical Report TR-520, Computer Science Center, University of Maryland, College Park, April 1977.

[Davis77b] L.S. Davis and A. Rosenfeld, "Curve Segmentation by Relaxation Labeling," *IEEE Trans. on Computers*, Vol. C-26, No. 11, pp. 1053-1057, November 1977.

[Davis77c] L.S. Davis, "Shape Matching Using Relaxation Techniques," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Troy, New York, pp. 191-197, June 6-8, 1977.

[Diamond82] M.D. Diamond, N. Narasimhamurthi, and S. Ganapathy, "A Systematic Approach to Continuous Graph Labeling with Application to Computer Vision," *Proc. National Conf. on Artificial Intelligence*, Pittsburgh, Pennsylvania, pp. 50-54, August 18-20, 1982.

[Eberlein76] R.B. Eberlein, "An Iterative Gradient Edge Detection Technique," *Computer Graphics and Image Processing*, Vol. 5, No. 2, pp. 245-253, 1976.

[Ekhundh80] J.O. Eklundh, H. Yamamoto, and A. Rosenfeld, "A Relaxation Method for Multispectral Pixel Classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 1, pp. 72-75, January 1980.

[Faugeras80a] O.D. Faugeras and M. Berthod, "Scene Labeling: An Optimization Approach," *IEEE Conf. on Pattern Recognition and Image Processing*, Chicago, Illinois, pp. 318-326, August 6-8, 1979. Also in *Pattern Recognition*, Vol. 12, No. 5, pp. 339-347, 1980.

[Faugeras80b] O.D. Faugeras, "An Optimization Approach for Using Contextual Information in Computer Vision," *Proc. 1st Annual National Conf. on Artificial Intelligence*, Stanford, California, pp. 56-60, August 18-21, 1980.

[Faugeras81] O.D. Faugeras and K. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," *IEEE Trans. on Pattern Anal. and Machine Intelligence*, Vol. PAMI-3, No. 6, pp. 633-642, November 1981.

[Fekete81] G. Fekete, J.O. Eklundh, and A. Rosenfeld, "Relaxation: Evaluation and Applications," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-3, No. 4, pp. 459-469, July 1981.

[Freuder76] E.C. Freuder, *A Computer System for Visual Recognition Using Active Knowledge*, Technical Report AI-TR-345, Artificial Intelligence Laboratory, Massachussetts Institute of Technology, Cambridge, Massachussetts, June 1976.

[Guzman68] A. Guzman-Arenas, *Computer Recognition of Three-dimensional Objects in a Visual Scene*, Technical Report AI-TR-228 (MAC-TR-59), Artificial Intelligence Laboratory, Massachussetts Institute of Technology, Cambridge, Massachussetts, (AD-692-200), December 1968.

[Haralick79] R.M. Haralick and L.G. Shapiro, "The Consistent Labeling Problem: Part 1," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 2, pp. 173-184, April 1979.

[Haralick80] R.M. Haralick, J.C. Mohammed, and S.W. Zucker, "Compatibilities and the Fixed Points of Arithmetic Relaxation Processes," *Computer Graphics and Image Processing*, Vol. 13, No. 3, pp. 242-256, July 1980.

[Haralick83] R.M. Haralick, "An Interpretation for Probabilistic Relaxation," *Computer Vision, Graphics, and Image Processing*, Vol. 22, No. 3, pp. 388-395, June 1983.

[Hart77] P.E. Hart and R.O. Duda, *PROSPECTOR–A Computer-Based Consultation System for Mineral Exploration*, Technical Note 155, Artificial Intelligence Center, SRI International, Menlo Park, California, October 1977.

[Hummel78] R.A. Hummel and A. Rosenfeld, "Relaxation Processes for Scene Labeling," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 10, pp. 765-768, October 1978.

[Hummel80] R.A. Hummel and S.W. Zucker, "On the Foundations of Relaxation Labeling Processes," Technical Report TR-80-7, Computer Vision and Graphics Laboratory, McGill University, Montreal, July 1980. Summarized in *Proc. 5th Int. Jnt. Conf. on Pattern Recognition*, Miami Beach, Florida, pp. 50-53, December 1-4, 1980.

[Kandel78] A. Kandel and W.J. Byatt, "Fuzzy Sets, Fuzzy Algebra, and Fuzzy Statistics," *Proc. IEEE*, Vol. 66, No. 12, pp. 1619-1639, December 1978.

[Kirby80] R.L. Kirby, "A Product Rule Relaxation Method," *Computer Graphics and Image Processing*, Vol. 13, No. 2, pp. 158-189, June 1980.

[Kirkpatrick83] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671-680, May 13, 1983.

[Kitchen80] L. Kitchen, "Relaxation Applied to Matching Quantitative Relational Structures," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-10, No. 2, pp. 96-101, February 1980.

[Lev77] A.Lev, S.W. Zucker, and A. Rosenfeld, "Iterative Enhancement of Noisy Images," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 7, No. 6, pp. 435-442, June 1977.

[Montanari74] U. Montanari, "Networks of Constraints: Fundamental Properties and Applications to Picture Processing," *Information Sciences*, Vol. 7, pp. 95-132, 1974.

[Nagin82] P.A. Nagin, A.R. Hanson, and E.M. Riseman, "Studies in Global and Local Histogram-guided Relaxation Algorithms," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4, No. 3, pp. 263-277, May 1982.

[O'Leary80] D.P. O'Leary and S Peleg, *Analysis of Relaxation Processes: The Two-node, Two-label Case*, Computer Vision Laboratory Technical Report TR-967, University of Maryland, College Park, November 1980.

[Peleg78a] S. Peleg and A. Rosenfeld, "Determining Compatibility Coefficients for Curve Enhancement Relaxation Processes," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 7, pp. 548-555, July 1978.

[Peleg78b] S. Peleg, "Iterative Histogram Modification, 2," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 7, pp. 555-556, July 1978.

[Peleg80a] S. Peleg, "A New Probabilistic Relaxation Scheme," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 4, pp. 362-369, July 1980.

[Peleg80b] S. Peleg, "Monitoring Relaxation Algorithms Using Labeling Evaluations," *Proc. 5th Int. Jnt. Conf. on Pattern Recognition*, Miami Beach, Florida, pp. 54-57, December 1-4, 1980.

[Quam78] L.H. Quam, *Road Tracking and Anomaly Detection in Aerial Imagery*, Technical Note 158, Artificial Intelligence Center, SRI International, Menlo Park, California, March 1978.

[Richards80] J.A. Richards, D.A. Landgrebe, and P.H. Swain, "Overcoming Accuracy Deterioration in Pixel Relaxation Labeling," *Proc. 5th Int. Jnt. Conf. on Pattern Recognition*, Miami Beach, Florida, pp. 61-65, December 1-4, 1980.

[Riseman77] E.M. Riseman and M.A. Arbib, "Computational Techniques in the Visual Segmentation of Static Scenes," *Computer Graphics and Image Processing*, Vol. 6, No. 3, pp. 221-276, June 1977.

[Rosenfeld76] A. Rosenfeld, R.A. Hummel, and S.W. Zucker, "Scene Labeling by Relaxation Operations," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-6, No. 6, pp. 420-433, June 1976.

[Rosenfeld77a] A. Rosenfeld and L.S. Davis, *Iterative Histogram Modification*, Technical Report TR-519, Computer Science Center, University of Maryland, College Park, April 1977.

[Rosenfeld77b] A. Rosenfeld, "Iterative Methods in Image Analysis," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Troy, New York, pp. 14-18, June 6-8, 1977.

[Rosenfeld78] A. Rosenfeld and L.S. Davis, "Iterative Histogram Modification," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 4, pp. 300-302, April 1978.

[Rosenfeld82] A. Rosenfeld, "Picture Processing: 1981," *Computer Graphics and Image Processing*, Vol. 19, No. 1, pp. 35-75 (esp. p. 59), May 1982.

[Rosenfeld83] A. Rosenfeld, "Picture Processing: 1982," *Computer Vision, Graphics and Image Processing*, Vol. 22, No. 3, pp. 339-387 (esp. pp. 368-369), June 1983.

[Schachter76] B.R. Schachter, A. Lev, S.W. Zucker, and A. Rosenfeld, *An Application of Relaxation Methods to Edge Reinforcement*, Technical Report TR-476, Computer Science Center, University of Maryland, College Park, August 1976.

[Troy73] E.B. Troy, E.S. Deutsch, and A. Rosenfeld, "Gray-Level Manipulation Experiments for Texture Analysis," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SMC-3, No. 1, pp. 91-98, January 1973.

[Ullman79] S. Ullman, "Relaxation and Constrained Optimization by Local Processes," *Computer Graphics and Image Processing*, Vol. 10, No. 2, pp. 115-125, June 1979.

[VanderBrug77] G.J. VanderBrug, "Experiments in Iterative Enhancement of Linear Features," *Computer Graphics and Image Processing*, Vol. 6, No. 4, pp. 25-42, April 1977.

[Waltz72] D.L. Waltz, *Generating Semantic Descriptions from Drawings of Scenes with Shadows*, Ph.D. Dissertation, Technical Report AI-TR-271, Artificial Intelligence Laboratory, Massachussetts Institute of Technology, Cambridge, Massachussetts, (AD-754-080), August 1972.

[Yamamoto79] H. Yamamoto, "A Method of Deriving Compatibility Coefficients for Relaxation Operators," *Computer Graphics and Image Processing*, Vol. 10, No. 3, pp. 256-271, July 1979.

[Yakimovsky73] Y. Yakimovsky and J.A. Feldman, "A Semantic Based Decision Theory Region Analyzer," *Proc. 3rd Int. Jnt. Conf. on Artificial Intelligence*, pp. 580-588, August 1973.

[Yakimovsky76] Y. Yakimovsky, "Boundary and Object Detection in Real World Images," *J. ACM*, Vol. 23, No. 4, pp. 599-618, October 1976.

[Zadeh65] L.A. Zadeh, "Fuzzy Sets," *Inform. Control*, Vol. 8, pp. 338-353, 1965.

[Zucker76] S.W. Zucker, "Region Growing: Childhood and Adolescence," *Computer Graphics and Image Processing*, Vol. 5, No. 3, pp. 382-399, September 1976.

[Zucker77] S.W. Zucker, R.A. Hummel, and A. Rosenfeld, "An Application of Relaxation Labeling to Line and Curve Enhancement," *IEEE Trans. on Computers*, Vol. C-26, No. 4, pp. 394-403 (and 922-929), April 1977.

[Zucker78a] S.W. Zucker, E.V. K[illegible]shnamurthy, and R.L. Haar, "Relaxation Processes for Scene Labeling: Co[illegible]rgence, Sp[illegible] and Stability," *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. SM[illegible], pp. 41-48, January 1978.

[Zucker76b] S.W. Zucker and J.L. Mohammed, "Analysis of Probabilistic Relaxation Labeling Processes," *Proc. IEEE Conf. on Pattern Recognition and Image Processing*, Chicago, Illinois, pp. 307-312, May 31 - June 2, 1978.