INTEGRATING NON-SEMANTIC KNOWLEDGE INTO IMAGE SEGMENTATION PROCESSES

Ralf R. Kohler

COINS Technical Report 84-04

This work was supported in part by the Office of Naval Research under grant number N00014-75-C-0459, the National Science Foundation under grant number MCS79-18209, and the National Institute of Health under grant number RR07048-16.

This work was supported in part by:

The Office of Naval Research
Grant Number N00014-75-C-0459

and

The National Science Foundation
Grant Number MCS79-18209

and

National Institute of Health
Grant Number RR07048-16

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to Dr. Edward Riseman and Dr. Allen Hanson for their continuous guidance and support over the many years this dissertation required and for their extensive editing of this dissertation. I would also like to thank Dr. Victor Lesser and Dr. Robert Huguenin, the other members of my committee, for their helpful comments and encouragement.

Secondly, I would like to thank the various members of the VISIONS research group who helped with the design and implementation of the image operating system (IOS) including John Prager, Bryant York, Paul Nagin, and Tom Williams among others. Also I thank Ken Ward and Nancy Irwin who helped in the implementation of some of the algorithms and running of experiments and Joey Griffith who implemented the region merging algorithm.

affect the performance of the process.

At the center of our segmentation system is an algorithm which labels pixels in localized subimages with the feature histogram cluster to which they correspond, followed by a relaxation labeling process. However, this algorithm has a tendency to undersegment by failing to find clusters corresponding to small objects; it may also oversegment by splitting intensity gradients into multiple clusters, by finding clusters for "mixed pixel" regions, and by finding clusters corresponding to microtexture elements. In addition, the relaxation process often destroys fine structure in the image. Finally, the artificial subimage partitions introduce the problem of inconsistent cluster sets and the need to recombine the segmentations of the separate subsimages into a consistent whole. This dissertation addresses each of these problems by adding and deleting clusters based on image space information, by merging regions, and by defining different compatibility coefficients in the relaxation so as to preserve fine structures. The result is a segmentation algorithm which is more reliable over a broader range of images than the simple clustering algorithm.

Solutions to the same segmentation problems were examined via the integration of different segmentation algorithms (including edge, region, and thresholding algorithms) to produce a consistent segmentation. Multi-process integration techniques varied from static integration of the final results generated by individual algorithms through dynamic integration of the processes themselves. The resulting unified segmentations from these approaches were generally better than segmentations

ABSTRACT

Integrating Non-Semantic Knowledge
into Image Segmentation Processes

September 1983

Ralf R. Kohler

B.S. Virginia Polytechnic and State University

M.S. University of Massachusetts

Ph.D., University of Massachusetts

Directed by: Professor Edward Riseman

This dissertation develops several techniques for automatically segmenting images into regions. The basic approach involves the integration of different types of non-semantic knowledge into the segmentation process such that the knowledge can be used when and where it is useful. These processes are intended to produce initial segmentations of complex images which are faithful with respect to fine image detail, balanced by a computational need to limit the segmentations to a fairly small number of regions.

Natural scenes often contain intensity gradients, shadows, highlights, texture, and small objects with fine geometric structure, all of which make the calculation and evaluation of reasonable segmentations for natural scenes extremely difficult. The approach taken by this dissertation is to integrate specialized knowledge into the segmentation process for each kind of image event that can be shown to adversely

produced by any of the constituent algorithms.

Finally, the dissertation also describes the Visions Image Operating System (IOS) which made all of the experiments in this dissertation possible. This software environment, driven by an interactive user interface in LISP, provides a powerful experimental tool in which complex image analysis algorithms can be easily integrated and applied to images of different structure and resolution. The IOS is currently being used by many image analysis researchers at the University of Massachusetts and at several other sites involved in industrial, remote sensing, and medical applications.

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Research Objectives

The problem of visual perception by computer is often split into two stages. The first stage of processing partitions or segments the scene into regions of differing visual characteristics, while the second stage attempts to build a three dimensional model of the scene using this segmentation, the image, and prior world knowledge. This dissertation focusses solely upon the problem of producing a reliable segmentation from a visually complex image and proposes a variety of mechanisms to overcome the limitations of existing segmentation algorithms. The basic methodology involves complementary sources of information and constraints to improve upon the limited information used by any single segmentation algorithm.

While the use of semantic information to guide the segmentation process has potential in specialized applications, it is our position that segmentation algorithms should probably not be based on knowledge of the objects to be recognized. The view taken here is that higher level processes, which utilize semantic knowledge, should influence or tune the low level segmentation processes, but semantic knowledge such as known object identities should not be utilized directly by the segmentation processes. This will, hopefully, result in a domain independent segmentation system which can easily be tuned to take advantage of prior knowledge for a given domain.

Natural scenes often contain intensity gradients, shadows, highlights, texture, and small objects with fine geometric structure, all of which make the calculation and evaluation of reasonable segmentations for natural scenes extremely difficult. Because most segmentation algorithms are based on a small set of heuristics which reflect both implicit and explicit assumptions about the image, no segmentation algorithm can be expected to produce good segmentations when the assumptions on which it is based are violated.

The approach taken by this dissertation is to integrate specialized knowledge into the segmentation process for each kind of image event that can be shown to adversely affect the performance of the process. This may be done by integrating the knowledge into a single segmentation algorithm, integrating segmentation algorithms based on different forms of image information, and integrating segmentations produced by algorithms based on different knowledge.

This dissertation contains five major components:

(1) A software environment, specifically designed for dynamic experimentation in image analysis, was implemented to facilitate the research presented elsewhere in the dissertation.

(2) Two extensions were made to the Nagin cluster based segmentation algorithm used in chapter 6. The extensions modified the decision boundary for initial classification of pixels and the compatibility coefficients used in the relaxation update algorithm.

(3) A paradigm for development of image segmentation algorithms, based on quantitative evaluation of segmentations, is proposed.

(4) A methodology for the intelligent integration of multiple sources of knowledge into a single segmentation algorithm is explored. In particular, the selection of clusters in the Nagin clustering and relaxation algorithm is improved using spatial expectations about the clusters.

(5) The integration of multiple segmentations and segmentation algorithms based on different knowledge and different image features is considered.

1.2 Global Context

In our research, segmentations are produced in the context of a complete image interpretation system, called VISIONS ([HAN75], [HAN78b]), which attempts to build a three-dimensional description from images of natural outdoor scenes. Figure 1 shows an overview of this system. In VISIONS, the segmentation executive builds an initial segmentation of the scene, which is then used by the image interpretation system to build a set of hierarchically structured hypotheses about the particular scene based on stored world knowledge. When necessary, these hypotheses about the semantic content of the scene can be used to produce feedback requests to the segmentation executive to modify or refine the segmentation. This implies that the initial segmentation need not be "ideal", but it must be sufficiently detailed to allow the interpretation system to extract general image properties in order to begin goal-directed processing. The primary emphasis of this dissertation is on the the startup problem of producing a "reasonable" initial segmentation without utilizing general semantic data or specific information about the contents of the image.

In the context of the VISIONS image interpretation system, the segmentation is an intermediate representation of the scene in which the raw sensory data has been organized into symbolic structures such as regions, segments, and vertices. Currently, there is a controversy whether a segmentation is a necessary or desirable intermediate structure, and whether there exists a physiological analogue of segmentation in human visual processing. However, human subjects can perceive large uniform areas even when presented with images which do not contain meaningful or familiar objects, or when inadequate context makes recovery of the semantic content of a scene difficult or impossible. This implies that semantic labeling of the objects in the scene is not necessary to produce a segmentation and supports the use of such an intermediate representation in the process of scene analysis.

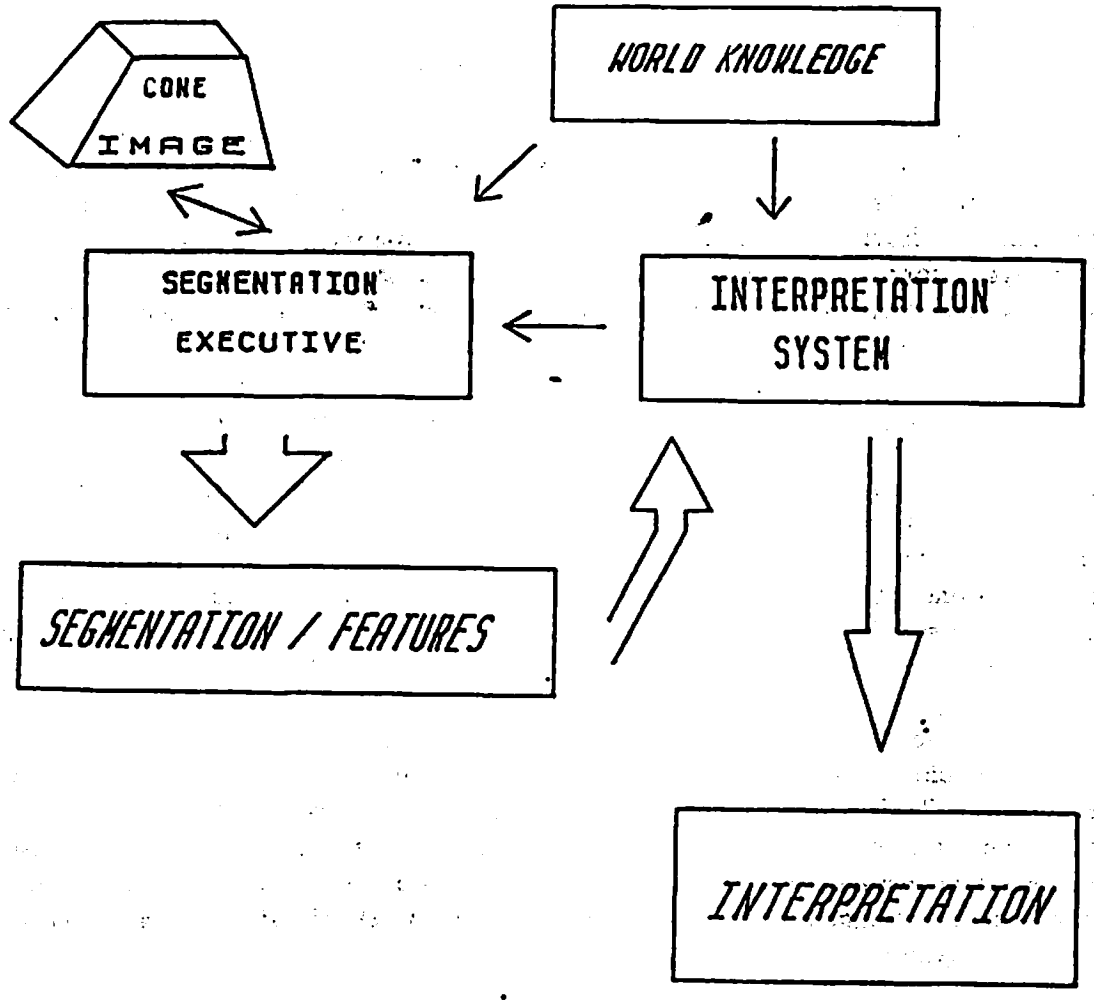


Figure 1: VISIONS System Overview.

1.3 Segmentation and Evaluation

A segmentation of an image is a partition of the picture elements (or pixels) into disjoint sets of spatially contiguous pixels (referred to as regions). The goal of the image segmentation algorithms is to produce segmentations for which there is a high correlation between the entities of the real world (objects, surfaces, and parts of objects) and the regions of the segmentation. That is, each region in the image should ideally correspond to at most one object part.[1]

It is difficult to overstate the complexity of the segmentation problem. In relatively complex, unconstrained scenes, such as full color outdoor scenes, any straight-forward approach is prone to gross errors. Intrinsic characteristics of the scene such as direct and indirect lighting, varying orientation of surfaces, shadows, texture, specularity, and noise in the segmentation system (especially due to the discrete digital representation) make the generation of "good" segmentations very difficult.

One of the goals of this dissertation is to examine potential improvements in segmentation processes by integrating multiple sources of image data and segmentation models into a unified algorithm and to evaluate these algorithms in a context which utilizes specialized preprocessing and postprocessing to enhance the performance of the algorithms. In order to evaluate control structures which involve alternative forms of algorithm interaction and the effect of a particular preprocessing step, one must be able to evaluate and compare the resulting alternative segmentations in a quantitative manner.

The problem of segmentation evaluation is an open problem. It is not generally possible to produce a "correct" segmentation of a complex natural scene, since a "correct" segmentation implies underlying assumptions about the goals of the segmentation process ([FIR72], [NAG80]). In this dissertation we discuss the issues of segmentation evaluation and propose an appropriate methodology for development and evaluation of segmentation algorithms based on a set of artificial test images of varying complexity.

1.4 The VISIONS Image Operating System (IOS)

Another difficulty with both segmentation algorithm development and algorithm evaluation stems from the computational cost of computing even a single segmentation for a high resolution image. A reasonable image might contain 512 by 512 pixels, with three colors and eight bits per color at each pixel, for a total of approximately six million bits. A particular algorithm might perform dozens of operations at each pixel, where each operation might consist of hundreds of machine

---

[1] A "good" segmentation, typically, would consist of regions such that each region would be relatively invariant in some image feature while each boundary between regions would exhibit a significant, often sharp, gradient in some image feature.

instructions. This computational cost makes it virtually impossible to perform a thorough examination of the parameter space in order to determine the effectiveness of interactions of multiple algorithm components. In order to facilitate the development and evaluation of complex segmentation algorithms, a programming environment for segmentation was developed which supports highly interactive and dynamic experimentation.

The VISIONS image operating system (IOS) was designed to allow interactive and flexible construction of segmentation algorithms out of efficient primitives (referred to as image operators). The IOS is based on the hierarchical "processing cone" model of Hanson and Riseman ([HAN74], [HAN80]). The IOS provides a friendly environment via LISP for the dynamic ,experimentation with, and development of, image segmentation algorithms. The IOS is described in detail in chapter 2. The IOS was designed and implemented by the author with considerable support from many other members of the VISIONS research group. Although the IOS greatly facilitated the experiments presented in this dissertation, it is not necessary to understand the IOS implementation in order to understand the remainder of the dissertation; thus chapter 2 may be skipped.

1.5 Segmentation Background

Existing image segmentation algorithms can be divided into two broad classes. The first class attempts to build regions in the image based on **similarities** of some characteristics (or features) of the pixels in the image. The second class can be viewed as implicitly building regions by locating those edges in the image which correspond to **differences** between pixel characteristics. One of the goals of this dissertation is to integrate these approaches such that both similarity and difference information is embedded in the segmentation process. Chapter 3 examines several segmentation algorithms with emphasis on the region and edge segmentation algorithms utilized in the remainder of the dissertation.

The region algorithm [NAG79] is based on an initial cluster analysis of image features in order to determine the likelihood that each pixel "belongs" to each cluster. After assigning cluster label likelihoods to pixels in image space, an iterative probabilistic relaxation is performed. The goal is to use the local context at a pixel to obtain a consistent labelling for each pixel. The edge algorithm [HAN78a] forms initial edge probabilities based on local contrast between adjacent pixels as input to a probabilistic relaxation algorithm. The updating operator has been developed from a Bayesian view of the edge patterns which are either consistent or inconsistent with the formation of continuous boundaries. A third algorithm used in this dissertation is a segmentation algorithm based on thresholding ([KOH78], [KOH81]). This algorithm uses edge contrast to make threshold selection decisions in a partial integration of the edge and region based approaches. Although these algorithms are the only major segmentation algorithms employed, the findings and methods of the dissertation should be easily extrapolated to other segmentation algorithms. Chapter 3 also compares this dissertation with other systems which have taken a relatively

complete approach to image understanding, as well as to systems which attempt to use multiple knowledge sources to attack similarly complex problems.

1.6 Segmentation Algorithm Extensions

Chapter 4 contains two extensions of the Nagin cluster based segmentation algorithm described in chapter 3. These extensions, while not integrally related to the theme of this dissertation, do improve the quality of the segmentations produced and enhance the robustness of the relaxation component. The first modification selects decision boundaries between clusters heuristically to minimize the number of errors remaining after the relaxation. The second modification altered the center pixel's contribution to the relaxation update such that the algorithm could be tuned to preserve desirable geometries while destroying others.

1.7 Segmentation Evaluation

Chapter 5 examines the difficult problem of segmentation evaluation. This chapter proposes a methodology for segmentation algorithm development. The approach applies algorithms to images of ever increasing complexity, beginning with very simple artificial images, for which quantitative evaluation is possible, and ending with complex natural scenes which contain all the image characterics which complicate the real world, but for which meaningful quantitative segmentation evaluation is not possible.

1.8 Integrating Knowledge into a Segmentation Algorithm

Chapter 6 develops the mechanisms for integrating different knowledge into an existing segmentation algorithm. This chapter focuses on the selection of clusters in the Nagin cluster based segmentation algorithm. We investigate the addition of clusters in a subimage based on consistency between the current cluster set and the cluster sets in adjacent subimages. This chapter also considers the deletion of clusters based on expectations about the spatial behavior of image regions formed by the clusters. In particular, the algorithm attempts to identify clusters which give rise to micro-texture, gradients, and "mixed pixel" regions. Finally, a region merging algorithm based on multiple sources of knowledge which could contribute to a merge decision, is presented. The merge algorithm provides a general framework in which many very different kinds of knowledge may be uniformly integrated into a segmentation process. This merging algorithm could be easily extended to utilize more complex region merge rules, including rules based on feedback from the semantic interpretation system or segmentation executive. This combined segmentation algorithm, which utilizes the components described in this chapter, is shown to provide better and more reliable segmentations.

1.9 Integrating Alternative Segmentations and Segmentation Algorithms

Chapter 7 investigates several alternative approaches to the implementation of cooperative communication between segmentation processes. This includes simple static interaction models where results of several algorithms are combined, as well as more complex iterative or dynamic interaction models. The goal is to use interprocess consistency to enhance the quality of the segmentation produced. In one example, the integration of several segmentations using the region merging algorithm of chapter 6 is shown to produce a segmentation superior to any of the contributing segmentations.

1.10 Summary

Chapter 8 summarizes the results of this dissertation concluding that various kinds of non-semantic knowledge can be effectively integrated into the segmentation process to not only increase the quality of the segmentations produced, but more importantly, to increase the reliability of the segmentation process. In chapter 6 we have successfully integrated knowledge about consistent spatial contexts and predictions about the expected behavior of feature clusters in the image into a single segmentation algorithm based on region labeling by histogram clusters. In chapter 7 we integrated several segmentation algorithms which are based on different assumptions about useful information in the image. We believe, it should now be possible to move toward building an effective segmentation executive, utilizing the the approaches, techniques, and strategies presented in this dissertation in order to dynamically select algorithms, parameters, merge rules, and image features, for particular images or parts of images, to obtain high quality segmentations.

2.0 THE IMAGE OPERATING SYSTEM

2.1 Overview of the IOS

The Image Operating System is a complete software environment, built on LISP, specifically designed for dynamic experimentation in scene analysis. The IOS was used for all of the experiments presented in the remainder of this thesis.

In order to carry out complex experiments in which various segmentation algorithms might interact, perhaps using different image features, an environment conducive to such experimentation was needed. This environment is provided by the Image Operating System designed and implemented by the author with assistance from many members of the VISIONS research group at the University of Massachusetts. This image operating system is based on a computational structure known as a "processing cone" proposed by Hanson and Riseman ([HAN74], [HAN80]). This structure is specifically designed for hierarchical parallel operations on two dimensional arrays and is related to the recognition cones of Uhr [UHR72],

the hierarchical data structures of Klinger [KLI76], the pyramids of Tanimoto [TAN75], the quadtrees of Rosenfeld [ROS71], [HAY74], [ROS80], the planning algorithms of Kelly [KEL71] and Price [PRI71], and the knowledge-directed analysis of Ballard, Brown, and Feldman [BAL78]. A survey of some of the uses of these types of structures is found in [BURT83], [TAN78], and [TAN80].

The next section describes the processing cone model and the advantages of this model for image analysis. The remainder of the chapter presents an overview of the Image Operating System and considers some of the design decisions used to arrive at this particular system. For a functional description refer to "The Image Operating System Users Manual" [KOH84].

2.2 The Processing Cone

The processing cone is a model for a hierarchical parallel array processing architecture proposed by Hanson and Riseman ([Han74] and [Han80]). Figure 2 depicts the processing cone structure. At each level of resolution, n, the cone contains 2^n by 2^n pixels, with a vector of values, V, stored at each pixel. The corresponding vector elements V_i for all pixels at a given level of resolution can be considered as a single two-dimensional entity; this slice across the cone is referred to as a plane. A color image, where each pixel contains a three-element vector of the red, green, and blue components, would be represented as a set of three planes in the cone. The cone is hierarchical and each pixel at level n has four unique descendants at level n+1 and exactly one ancestor pixel at each level n-1 ($n >= 1$). If the raw data is at level m, m>n, then one can view a pixel at level n as having a receptive field of 2^{m-n} by 2^{m-n} pixels at level m.

Image operators are functions which are evaluated on a set of argument planes and which produce one or more output planes. There are three classes of operations which may be performed in the cone structure: iteration, reduction, and projection. For iteration, the level of resolution for both the domain planes and the range plane is the same. Iteration uses values in the spatial neighborhood of a domain pixel to produce new plane values at the corresponding pixel in the range planes. For reduction, the level of resolution of the range plane is less than the level of resolution of the domain planes. Reduction uses values in the neighborhood of the descendant pixels in the domain plane (level n+1) to produce a new plane value at the ancestor pixel in the range plane (level n). For projection, the level of resolution of the range plane is larger than the level of resolution of the domain plane. Projection uses information from a neighborhood about the unique ancestor pixels in the domain planes (levels m, $0=<m<n$) to produce values at each descendant range pixel (at level n). These basic operations may of course be combined in various ways to form more complex algorithms.[2]

[2] The implementation of the IOS actually allows for the definition of image operators which produce sets of output planes. This is mathematically equivalent but more convenient and more efficient.

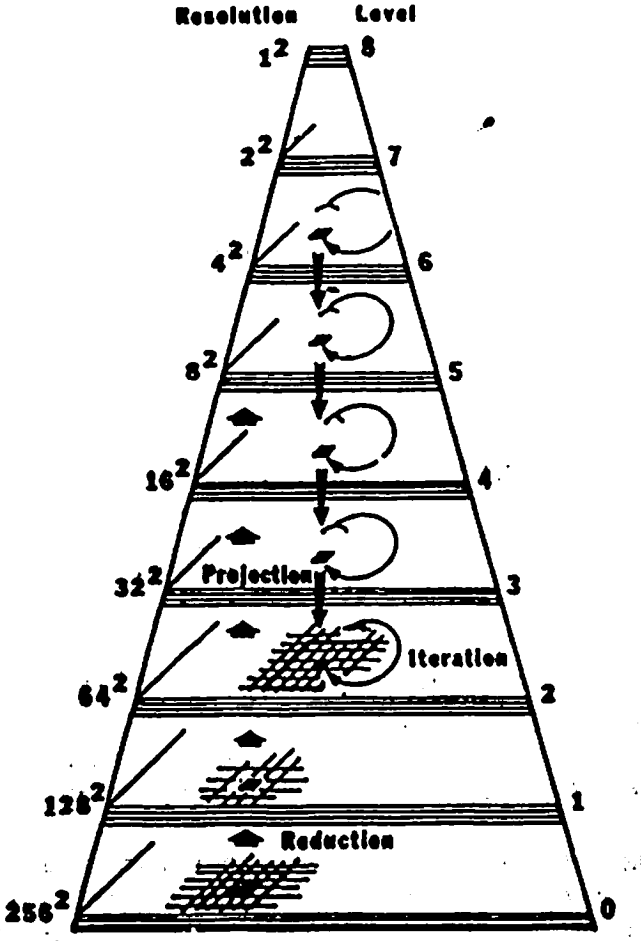


Figure 2: Cone Array Processor Structure.

Local image operations are defined by simple prototype functions on the neighborhood of a pixel to produce values in the output plane(s). Note that the output planes may be distinct from any of the input planes; e.g. an intensity image might be computed by using the original red, green, and blue input planes to produce a new intensity output plane. Figure 3 illustrates graphically three simple examples corresponding to the three different types of image operators:

(1) creating a "median" image by applying a three by three median filter is an example of iteration,

(2) creating a lower resolution "maximum" image by finding the maximum of all descendants of a pixel is an example of reduction,

(3) and finally, creating a locally thresholded output plane where the thresholds are found in a plane of lower resolution than the data is an example of projection.[3]

Since in general it is not known which operators are needed for particular applications and/or experimental investigations, a mechanism for dynamically specifying additional prototype functions (referred to as image operators) must be a part of any implementation of the cone model.

The cone structure has been shown to be useful for a host of image analysis problems. The general goals of image analysis include the transformation of a large spatial array of picture elements (pixels) into a more compact description of the image in terms of visually distinct syntactic units and their characteristics. The visual information in the image must be aggregated and labeled with symbolic names and attributes. The syntactic units most often used are boundary segments (connected sets of edges) and regions (connected sets of pixels), but other units are possible. The characteristics of boundaries include but are not limited to location, length, contrast, and orientation while region characteristics include size, shape, location, color, and texture. Image operators for computing all of the above have been implemented with the VISIONS Image Operating System.

The processing cone seems to be an appropriate model or architecture for rapidly performing the kinds of processing needed to implement generalized image processing operations for a number of reasons. The first consideration is that the processing cone is a parallel array architecture that is particularly suited to the enormous computational demands of image analysis. With images of reasonable spatial resolution (512 by 512 pixels) and reasonable color resolution (3 colors, 8 bits per color) about six million bits of data must be processed for each image. Many image segmentation algorithms (such as the relaxation algorithms described in chapter 3) require many iterations over this data to produce a final segmentation. The vast amount of data to be processed and the eventual constraint of real-time processing

[3] Actually this is an example of a combination of projection and iteration since one input plane is of lower resolution than the data while the other input plane is the data to be thresholded. This operator could have been split into a strict projection of the thresholds followed by a strict iteration operator to perform the thresholds.

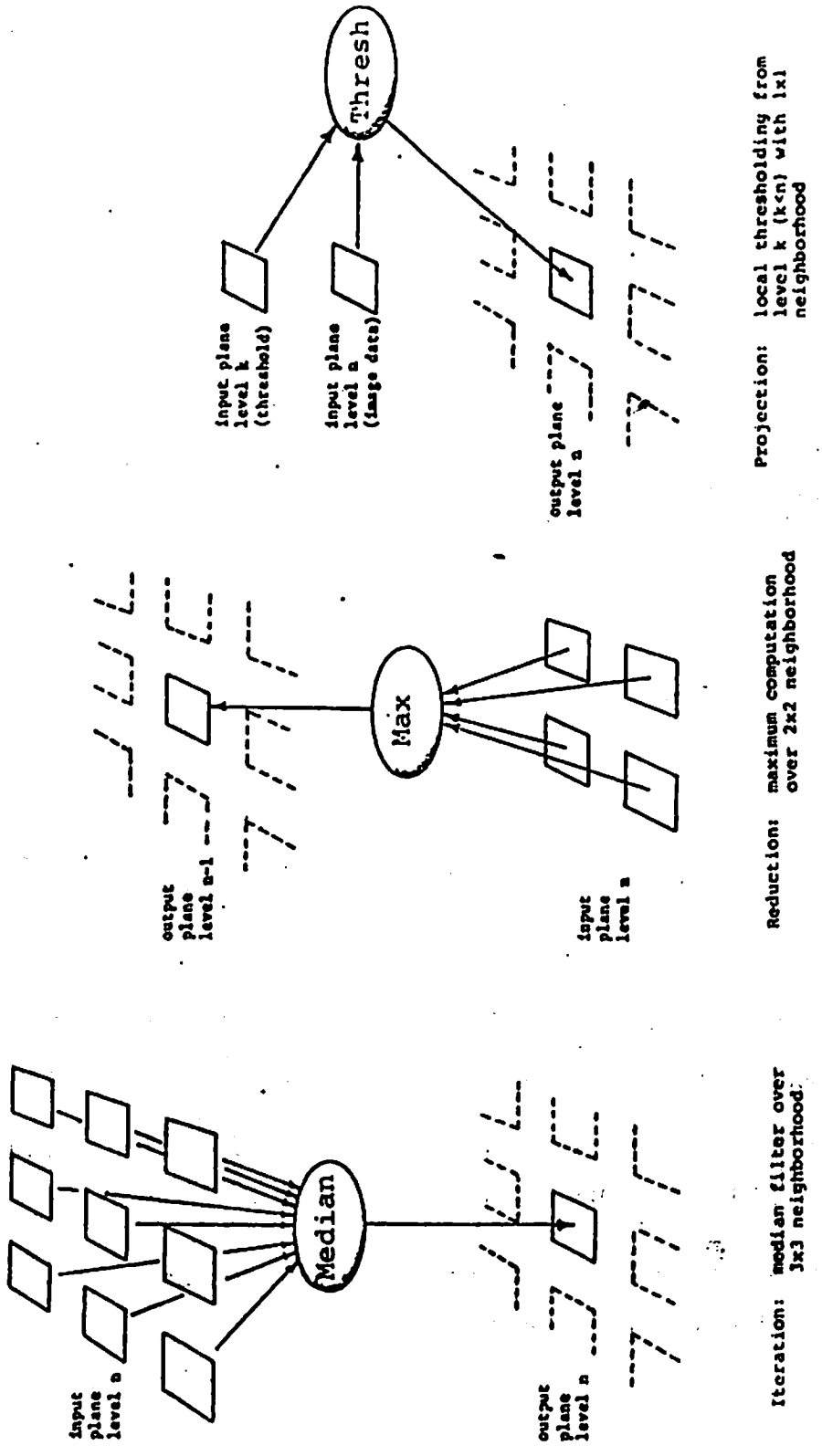


Figure 3: Projection, Reduction, and Iteration in the Cone.

indicate the necessity of a parallel processing approach.

A second consideration is that image operators can be simply defined. Image operations within the cone are defined locally by a prototype function which uses information at neighboring pixels to produce the result at the central pixel. The image operator is defined as the parallel application of the prototype function to all pixels in a plane. The parallel approach is especially viable now that advances in hardware technology may make such array processors economically feasible [Kru80], [Duf81]. The prototypical functions provide a mechanism for developing parallel algorithms. Combinations of these image operators applied within a suitable control structure are used to implement complex algorithms needed for segmentation and feature extraction.

The cone model also supports inherently hierarchical computations. The cone structure is very appropriate for aggregation of characteristics at coarser (lower) levels of resolution and for planning, where low resolution results are used to focus processing at finer (higher) levels. One recent algorithm which uses the hierarchical capabilities of the cone is a hierarchical image registration algorithm due to Glazer, Reynolds, and Anandan [GLA83].

A Users View.

One of the goals of the Image Operating System is to provide a flexible interactive environment in which a user can easily perform experiments involving many image operators and images at various levels of both spatial and color resolution. The system is designed as a research tool for algorithm development of parallel image processing as opposed to a production image processing system. A production system must heavily emphasize efficiency while a research system may favor flexibility, extensibility, and functionality over efficiency. This is necessary since the function to be performed by the research system is generally not as precisely defined. It should be straightforward to generate an efficient production system from the research system once the set of image operations to be performed in a production environment are determined.

In order to model the processing cone the Image Operating System provides:

(1) the data structures necessary to implement a generalized hierarchical cone structure,

(2) mechanisms for defining prototypical functions (image operators),

(3) methods for applying image operators,

(4) methods for specifying variable and plane bindings for image operator parameters and logical planes, and

(5) mechanisms for composing sets of image operators into complex algorithms.

The system should provide a "friendly" environment for both experienced and naive users. The simplicity with which new image operators can be defined and added to the IOS is one of the critical strengths of this system. An image operator may be specified without having to consider the spatial resolution of the images to which it might eventually be applied for three reasons:

(1) operators are written such that (at least conceptually) the operator is defined locally for parallel execution at each pixel,

(2) references to image neighbors, ancestors, and descendants are all referenced through a pixel centered coordinate system,

(3) and, references to pixels outside the physical image are handled automatically based on the users choice of access functions.

Furthermore, images of different data types and ranges can generally be handled without any extra code in the user function since all access to the image is through special access functions.[4]

Image operators are written as FORTRAN subroutines with four sections. The two most important sections are the **procedure** section, which defines the computation to be performed by the image operator, and the **environment** section, which provides the logical association (bindings or channels) between the logical planes utilized in the process section and the actual image planes resident in the execution data base (see below). Figure 4 shows a simplified example of a user function which computes the vertical and horizontal edge strength (i.e. a simple difference in values of a pair of horizontal and vertical pixels respectively) at each pixel. The input to the operator is simply an intensity plane and the output consists of a horizontal edge strength plane and a vertical edge strength plane.

A pixel-centered coordinate system and correct system handling of boundary conditions greatly simplify the specification of the prototype image operator since the image operator generally does not need to compute subscripts for neighboring pixels or consider special cases for neighboring pixels which fall outside of the image. The elements in the neighborhood are referenced via a pixel-centered coordinate system; the center pixel is at row 0 and column 0, the right neighbor is at row 0 and column 1, while the neighbor below is at row +1 and column 0. Note that the image operator specification is independent of the level of resolution of the data to which it will be applied and the data type of the input plane does not need to be specified. Such image operators can be easily added to the system for any function desired by the user.

---

[4] Real and Integer type operations are often distinguished both for reasons of efficiency and because FORTRAN distinguishes these types. Image operators would not distinguish between image planes whose type is logical, signed byte, unsigned byte, 16 bit integer, or 32 bit integer.

```
      Subroutine EDGES

      Parameter Intensity=1            ! intensity input plane
      Parameter Horizontal_edg=2       ! horizontal edge output plane
      Parameter Vertical_edg=3         ! vertical edge output plane
      Parameter Resolution_level=0     ! the relative level of resolution

c
c     Compute a one by two edge operator over the
c     intensity input plane.
c     The output consists of two planes:
c             The horizontal and vertical edge strengths.
c

      entry EDGES_environment          ! define the environment in which
c           ===================        ! the operator edges works
c
c
      call declare_input_plane( intensity, resolution_level)

c                                      ! the input plane determines the
c                                      ! absolute level of resolution

      call declare_output_plane( horizontal_edges, resolution_level)
      call declare_output_plane( vertical_edges, resolution_level)
c
c                                      ! the output planes will be defined
c                                      ! at the same level of resolution
      return
c
c
      Entry EDGES_process              ! This is the actual computation
c           ===============            ! of the operator at each pixel.
c
c
c           +------+
c           !r: -1!          Definition of neighborhood addressing
c           !c:  0!          r - relative row  c - relative column
c     +-----+------+------+
c     !r: 0 !r:  0!r:  0!
c     !c:-1 !c:  0!c:  1!
c     +-----+------+------+
c           !r:  1!
c           !c:  0!          The arguments to get_window_value are
c           +------+         (1) The plane from which to get a value
c                            (2) The relative row
c                            (3) The relative column
c

      center= get_window_value( intensity , 0, 0)     ! intensity at pixel
      right=  get_window_value( intensity , 0, 1)     ! intensity to right
      below=  get_window_value( intensity , 1, 0)     ! intensity below
c
      horiz_edge= center - below
      vert_edge=  center - right
c
      call set_window_value( Horizontal_edg, 0, 0, horiz_edge)
      call set_window_value( Vertical_edg, 0, 0, vert_edge)
c
c                                      ! the arguments to set_window_value
c                                      ! are identical to get_window_value
c                                      ! with the addition of the value
c                                      ! to set into the plane
c
      return
```

Figure 4: Example Prototype Image Operator.

Embedded within the operating system are mechanisms by which the user can write simple control functions to compose operators or to control the actual application of the operators to specified levels in the cone. The control functions are written in LISP. The programming language LISP is particularly well suited for an experimental environment, since it is an interpreted language and can therefore be modified interactively.[5] Figure 5 shows a simple example of control functions for an experiment using an edge relaxation system (see Chapter 3). The function EDGE-RELAX-5 computes the initial probabilities of edges from the intensity plane (which was provided in the call) and performs five iterations of edge relaxation. This function can then be used in an experiment (TEST-EDG-RELAX) which tries to evaluate the impact of the parameter MAXEDG (used in the computation of the initial edge probability) on the overall performance of the algorithm.

The user can look at the result of experiments in various ways: by printing the actual values found in the result planes, by printing a symbolic edge representation, by displaying direction encoded edge intensities on the graphic display device, or by displaying the original data as a surface whose height is proportional to the intensity (Figure 6). The ability to interact with the experimental algorithms and the ability to view the data in many different modes is extremely important in an interactive experimental environment.

An Implementors View.

The massive amount of data to be processed in image analysis was the overriding consideration in the design and implementation of the Image Operating System. The system must apply operators to images whose size might be as large as 2^{24} pixels (more than 16 million pixels). This requirement implies that efficiency is an important goal in the design of the system. However, as we have pointed out, since the system is to be used in a research environment, flexibility, functionality, and ease of use are also extremely important design considerations.

The system was designed and implemented to run on a VAX-11/780 host, utilizing the virtual memory management provided by the VMS operating system. In order to improve efficiency, some of the routines are coded in VAX assembly language (MACRO). In order to enhance functionality, VAX FORTRAN capabilities, which are not part of ANSI Standard FORTRAN, were used in many routines. In particular, VAX array and string descriptors were dynamically manipulated so that arrays and character strings could be allocated dynamically by the image operators at execution time allowing for more generalized image operators. Thus, transporting this system to a computer without virtual memory would require considerable recoding and development of software predictive paging mechanisms.

---

[5] The choice of LISP as a control language is justified in more detail in the section on the control language below.

```
(setq  my-house  (GET-PLANE  'mycone  'house  7))

!       Get an image with 128 pixels on a side (2**7) from
!       plane 'house' in cone file 'mycone' calling it my-house
!

(defun get-initprbs (intensity)
        (INITPROBS (EDGES (intensity))))

!       This code defines a function called get-initprbs
!       which computes initial edge probabilities by
!       applying the image operator INITPROBS to the result
!       of the image operator EDGES (defined in figure 4)

( defun Edge-relax-5  (intensity)
        (EDGERELAX (EDGERELAX (EDGERELAX (EDGERELAX
                (EDGERELAX (get-initprobs intensity)
                                        ))))))

!       Edge-relax-5 computes five iterations of edge relaxation
!       on the result of get-initprobs applied to intensity
!       (more powerful control constructs are available for both
!               iteration and recursion in LISP)

( defun Test-edg-relax (intensity maxedg)
        (set-parameter 'initprobs-maxedg  maxedg)
        (SHOWEDGES (edge-relax-5 intensity )'((maxval 1.0)))

!       Test-edg-relax applies the edge relaxation algorithm
!       for five iterations. The function INITPROBS uses a parameter
!       named "initprobs-maxedg" as a threshold.  Edges produced by
!       EDGES are converted to probability 1.0 if they excede
!       "initprobs-maxedg".  The function Test-edg-relax uses its
!       second argument as that parameter   SHOWEDGES displays the
!       result. The parameter "maxval" is provided to SHOWEDGES
!       such that a probability of 1.0 is scaled to full brightness
!       on the display device.

(fc-cons 'Test-edg-relax (* my-house) '( 5 10 20 30 40 )))

!       Now perform the experiment using five different values of
!       maxedg  The set of all the results is returned by the function
```

Figure 5: Experiment Using Image Operator Sequences.

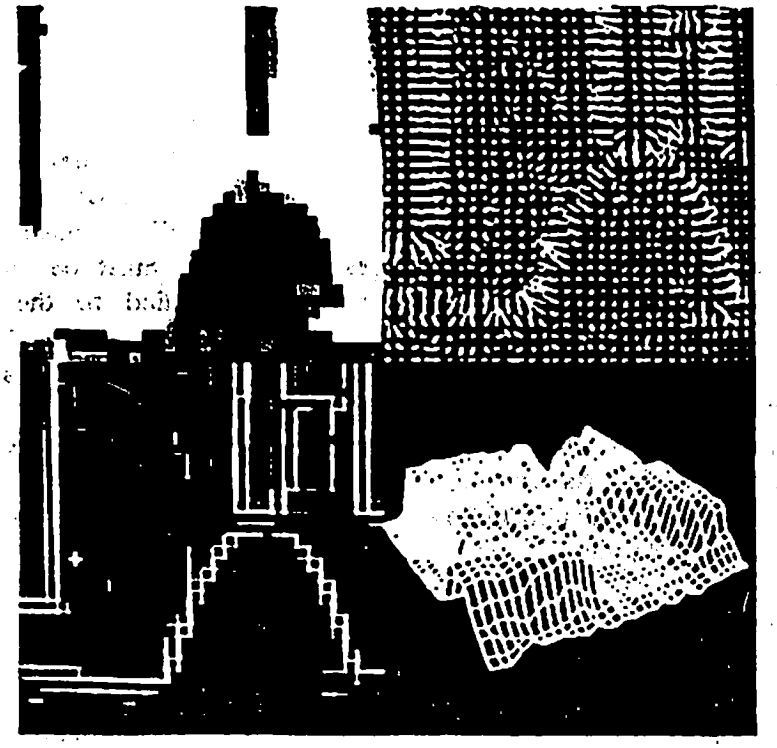


Figure 6: Alternative Displays of Intensity Change.

Figure 7 shows the overall structure of the system. Each of the major components of the VISIONS IOS is discussed separately below. The user communicates with the system via LISP functions and an auxiliary menu system. The operator application executive retrieves the user-selected image operators from the image operator library and applies the operator to the user-selected image in the execution data base.

In order to have the frequently called image accessing functions be as efficient as possible, the images to which image operators are applied must reside in the paging memory of the host in a form allowing for very rapid access. This data base of images is called the Execution Data Base (EDB). Images to be stored between executions of the IOS are saved in the Image Data Base in a form which minimizes the storage cost rather than the access time. The Image Data Base is used only for the long term storage of images and images must be moved into the Execution Data Base before image operators can be applied to them. Similarly, computed results can be saved by copying them from the Execution Data Base into the Image Data Base. Note that the system described here is still under development; features which are incomplete or unimplemented are so noted below. However, the system that does exist has been effectively used for several years at the University of Massachusetts and at several other sites to perform a number of different image processing tasks including industrial applications, medical applications, and remote sensing applications.

2.3 User Interface

The user interface is one of the most important components of an interactive system. The IOS provides a friendly environment for the dynamic definition of experiments with various images and image operators. The system contains comprehensive on-line documentation and prompting mechanisms, automatic record keeping mechanisms, and powerful error trapping and reporting mechanisms.

Control Language.

LISP was adopted as the language for interfacing the Image Operating System with the user. This was a natural choice for a number of reasons:

(1) LISP has all the powerful control structures that are needed.

(2) LISP is interpretive and therefore permits the dynamic generation of experiments which would otherwise require a compilation and link step in a non-interpretive language.

(3) LISP would provide a uniform interface with the semantic interpretation component of the VISIONS system since the interpretation system is based on a graph processing language called GRASPER, which is, in turn, built on LISP.

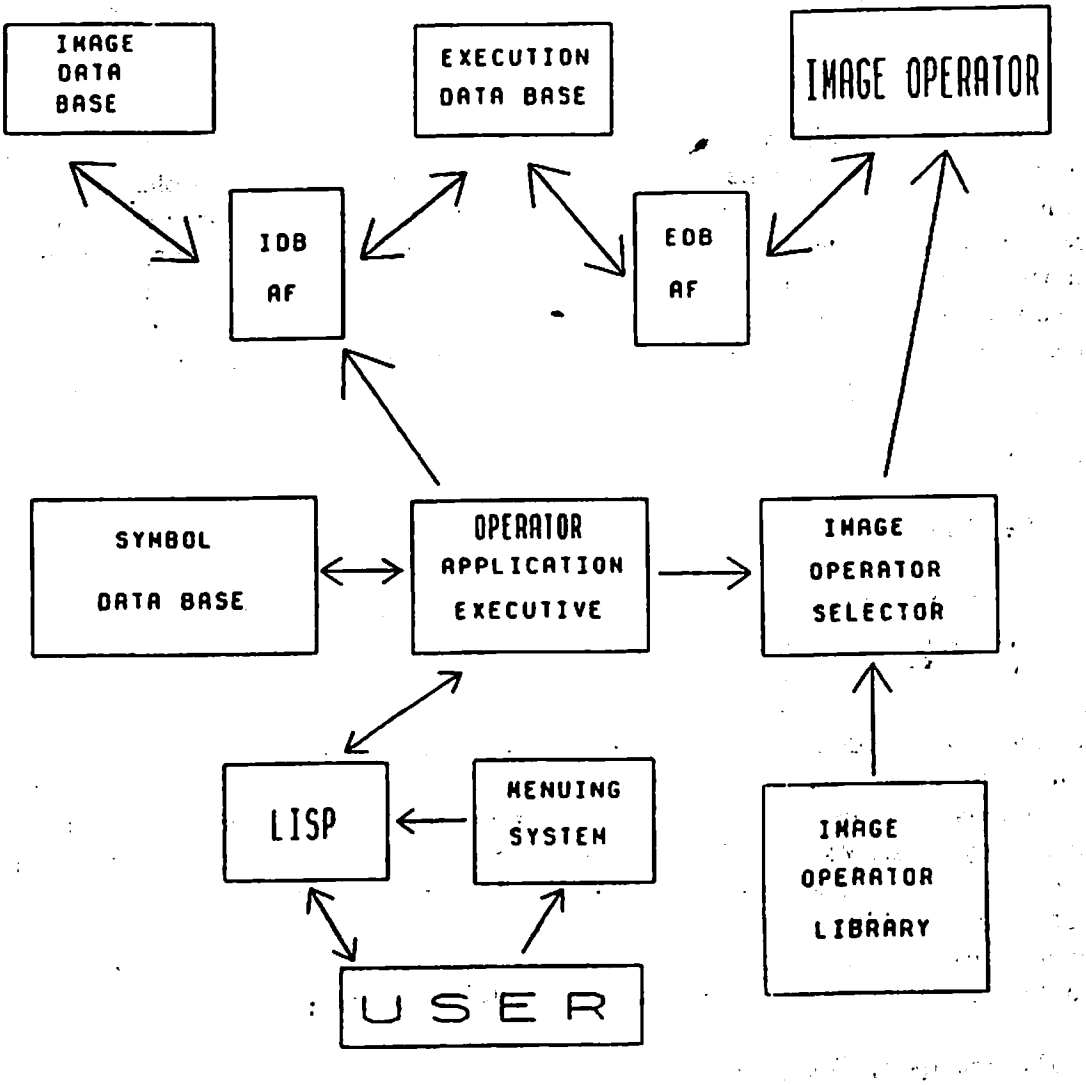


Figure 7: Image Operating System Overview.

The use of LISP for the entire implementation has the desirable quality of unifying the system implementation and making the portability of the system much more feasible. However; for reasons of efficiency a non-interpretive language was chosen for implementation of the underlying Image Operating System and the image operators (FORTRAN and VAX-MACRO). Thus, LISP provides flexible user control and access to the remainder of the underlying system. The interpretive overhead of LISP is not a liability in the IOS since the vast majority of the processing cost occurs in the FORTRAN and VAX-MACRO image operators.

Documentation.

This section outlines the documentation tools provided by the VISIONS IOS. In any system which supports multiple users of varying degrees of sophistication, it is extremely important to provide on-line documentation and documentation tools. Since the system is designed to be extended by new image operators, any external documentation will soon be out of date. Furthermore, the effort required to manually document the processing applied to an image is considerable if replication of the results at a later time is necessary- perhaps on another image. In the Image Operating System these two problems are handled by four documentation subsystems: the help subsystem, the prompting subsystem, the menu subsystem, and the history subsystem.

The Help Subsystem.

The help subsystem provides on-line help for any component of the system, any image operator included in the basic system, any image operator added by the user, many error conditions with their possible causes, and examples of image operators. The help system is hierarchically structured such that only the desired information is provided in response to a simple help request. The help files are structured to match the external VAX system documentation structure so that the help requests can be processed either within the Image Operating System or externally by VAX-VMS. There is one copy of the system help files which is shared by all users. Each user has unique help files for the image operators which are not shared. The help files have a straightforward structure and so can be easily modified by any user; furthermore, the help file for any new image operator added to the system by the user is automatically linked into the help tree.

The Prompting Subsystem.

The prompting subsystem provides a mechanism by which the necessary information for invoking a particular image operator can be obtained. This system permits the explicit specification of the necessary information before invocation of the operator, the automatic use of defaults for information not specified explicitly, and dynamic prompting for information when the default cannot generally be determined apriori. The use of this scheme permits a novice user to simply invoke an operator and be prompted for only the crucial information needed by that image

operator. An experienced user can fully control the image operation by specifying parameters which would otherwise be assigned default values. Note that all information which may be specified for a given image operator should be documented in the help system.

The Menu Subsystem.

The menu subsystem supports the development of comprehensive hierarchical menus which can be used to either sequence control directly or to build plans which can be executed at a later time. The menu system allows even naive users (without detailed knowledge of LISP or FORTRAN) to perform image processing experiments which fall within the domain of some predefined menu.[6]

The History Subsystem.

The history subsystem automatically maintains a complete history of the processing applied to a given image. All operations applied to all of the ancestor images of the current image are maintained in an ordered tree. For each operator, all parameter values are kept such that the operation might be replicated. The ordered descendants correspond to the ordering of the input images to the operator. These histories are compressed and saved with the image. This permits the user to know exactly how a given result was obtained. It is also possible to trade time against disk space by saving only the history of the image and recomputing the image when it is needed. It may be desirable to apply a sequence of operations to a small subimage to verify the correctness of the sequence of operations in real time and then use the generated history as a plan to automatically apply the sequence of operations to the whole image or a different image. Manually maintaining such detailed processing records is barely feasible, yet a failure to keep such records may make replication of the result at a later time virtually impossible.[7]

Error Handling.

The Image Operating System reports errors and warnings with appropriately descriptive messages. The ongoing operation continues for warnings, but control is returned to LISP for severe errors. It should be impossible to cause processing errors in the base system which result in termination of the Image Operating System. In order to extend this protection to image operators added by the user, an exception handler as provided by VAX-VMS is included in the function application executive. This handler allows a graceful recovery back to LISP after any unexpected errors within the user image operator. Note that expected errors (i.e. conditions for which the user operator tests) would be signalled directly as a warning or error.

---

[6] The menu system is fairly completely designed, but is, as of yet, unimplemented.

[7] The history system is fairly completely designed but unimplemented.

When a user operator does fail, the on-line help system actually permits the user to examine the source code or the listing file (if one exists) to find the code which produced the error. Often, the VAX-VMS dynamic debugger can then be used to temporarily patch the function (for the duration of the run) without ever exiting from the Image Operating System.

2.4 Data Bases in the IOS

The Image Operating System contains four different data bases. The image data base (IDB) is used for the long-term storage of images. The execution data base (EDB) is used for the short-term storage of images necessary for, or images produced by, image operators. The symbol data base (SDB) serves to communicate control information and non-image information between components of the system. The fourth data base is the library of image operators (IOPDB).

As can be seen in Figure 7 accesses to the data bases are made via specialized accessing functions. These accessing functions help to make the system modular. Changes in the implementation of any data base in the system affect only that data base and the corresponding accessing functions. Redefining the external structure of any of the accessing functions can, however, have repercussions throughout the system since all calls to the accessing function would have to be modified.

The Image Data Base (IDB).

The image data base is a hierarchical data structure for the long term storage of images at various levels of resolution. The data base is organized to minimize storage requirement and for convenient indexing for retrieval. The images (often referred to as "image planes" or just "planes") are indexed by cone name, image name, and level of resolution. A cone is a set of related images stored together in the same file. Each cone is organized into multiple levels of images; images within a level have the same spatial resolution. Finally, the image name specifies a particular image in the cone at that level of resolution.

The level of resolution is a power of 2 such that images of maximum size 2^n by 2^n pixels can be stored at resolution level n. The actual size of the image is stored with the image and only that portion of the image is actually maintained in the cone file. The cone structure also supports several different data types for the images. These data types include binary-valued images (1 bit per pixel), signed byte-valued images (8 bits per pixel), unsigned byte-valued images (8 bits per pixel), half-word-valued images (16 bits per pixel), integer-valued images (32 bits per pixel), and real-valued images (32 bits per pixel in a floating point format). The use of appropriate data types and subimage specifications permits the user to control both the spatial and magnitude resolution of the saved images and hence the disk space required to store them. On a system where disk space is scarce, images might be stored with only the number of bits actually required to store the image. However,

space conservation requires a conversion whenever data is transferred between the IDB and the EDB, with the reduction in performance implied by such a coding scheme. Therefore, only the data types used by the EDB are currently supported in the IDB.

For each image in the IDB, a description of the image and its processing history are saved with the image. The histories are variable length structures (LISP s-expressions) which are not dependent upon the particular implementation of LISP.

The IDB accessing functions (IDB-AF) contain functions which transfer images between the IDB and the EDB. The accessing functions also include user interface routines for querying the contents of the IDB cones, creating and deleting cones, and for deleting images (planes) within cones.

The Execution Data Base (EDB).

The execution data base (EDB) contains images which are input to image operators and images which are output by the image operators. The data base is organized to minimize access time for each pixel in an image being referenced. Images must be moved from the IDB into the EDB before image operators may access the image. All images in the EDB are exactly 2^n pixels by 2^n pixels. This permits pixel address calculations without multiplications. The EDB is allocated in the virtual memory space of the Image Operating System so that no software paging is needed. Since most image operators need only a small local context to produce the output values at each pixel and the IOS traversal of the image matches the storage of the image, very low page fault rates are generally obtained. Predictive software paging is appealing in such an image processing environment, but typically would result in a loss of functionality (via restrictions on the order of image traversal or on the size of the context), or would result in considerable software overhead if functionality is not sacrificed. Note that this decision limits the applicability of the current EDB to virtual memory systems such as the VAX.

Currently, a large EDB is allocated apriori, but dynamic extension of the EDB is possible. Current space management tools permit dynamic allocation, deletion, and coalescence of the space within the EDB.

When an image operator is applied, the logical image operator input and output planes are bound to a physical plane in the EDB such that the IDB-AF called by the image operator can access the image with minimum overhead.

The Symbol Data Base (SDB).

The symbol data base is used to pass control parameters and non-image data between components of the system. The default variable assignments and prompting for user-supplied variables are implemented within the SDB and its accessing functions. The symbol data base supports a number of data types (logical, byte,

half-word, integer, real, and character string) as well as multi-dimensional arrays of these primitive data types. The SDB provides the mechanisms for the dynamic allocation of these non-scalar parameters (including their descriptors). The SDB uses VAX descriptors for character strings and arrays for compatibility with other VAX-VMS software.

It would have been possible to implement the symbol data base directly in LISP. This would have the advantage that the LISP memory management tools could be utilized, simplifying maintenance of the tables. Also, temporary bindings for symbols would pose no special problems. That approach would also minimize the cost of accessing the symbols in LISP. A number of factors led to the decision to implement the symbol data base outside of LISP. LISP was not available when the symbol data base was first needed and modularity favored development of the SDB as a separate entity. Furthermore, the advantages of LISP memory management are lost for data types not directly implemented in LISP and the version of LISP available did not support any of the array data types discussed above. The saving of array contexts would also not be handled correctly for temporary bindings since LISP normally saves the context by simply allocating a new cell which points to the structure. This would not be adequate since the arrays are generally manipulated destructively. The solution of copying the entire structure would be expensive in both time and space. Saving the parameter context is possible in the current implementation but must be performed explicitly by the user.

2.5 Application of an Image Operator

The image operators utilize a set of input images and a set of operator-dependent parameters to produce a set of output images. Most operators are defined locally and the image operator application executive applies that local definition of the operator at each pixel in the image. Although this application is actually performed sequentially, there are no theoretical reasons why the computations could not be performed in parallel at all pixels, given the appropriate hardware implementation of the cone architecture.

Image operators are composed of four sections: the **environment** section (E$ section), the **initialization** section (I$ section), the **procedure** section (P$ section), and the **termination** section (T$ section). The environment section is executed once to define the context within which the operator will run. In this section channels are opened to access the input and output planes for the image operator. Parameters necessary for the computation of the operator are obtained and the operator declares itself and its purpose to the system. In short, the environment needed for the application of the operator is generated.

The initialization section is used to initialize any internal variables used by the operator. The process section contains the actual definition of the operator. Finally, the termination section performs any necessary cleanup (i.e. closing files) and sets any output parameters.

The process section obtains the values at neighboring pixels via functions which access the execution data base. In order to keep the definition of the operation as simple as possible, these accessing functions do not fail when pixels outside of the image are referenced. The accessing function instead returns either zero or the closest actual image value depending on the accessing function used. For instance, an operator which computes at each pixel the mean of the 3 by 3 neighborhood centered at the pixel would utilize an accessing function which returns the nearest image values for pixels outside of the actual image. The process section does not need to consider pixels at the periphery of the image as special cases and in fact no conditional instruction is needed in the process section of this simple operator.

The image application executive handles the proper invocation of these four sections for the operator selected by the user. The user can specify the subimage over which to apply the operator as well as the sampling density across the subimage (which would be desirable for implementing a non-overlapping window of a reduction operator).

2.6 Summary

The Image Operating System is one of the most advanced systems available for image processing today and represents a major, continuing, software development effort in the VISIONS research group. The system has been operational for several years at U. Mass. and three other sites. The IOS has been successfully applied to a number of image domains including natural scenes, biomedical images, Landsat images, robotics applications, and motion analysis.

The image operating system is a framework within which image processing algorithms and tools may be developed. During the IOS development the system evolved and was improved both by extensions to the actual IOS and by development of a pool of shared image processing functions. The majority of the software development effort has been expended toward the development of a shared library of documented, general-purpose, image operators. The library currently contains display drivers for different display formats on various display devices, image editing operators, statistical and feature extraction operators, generalized convolution operators, various contrast manipulation operators, noise reduction operators, edge enhancement operators, clustering operators, and classification operators.

In a recent text entitled "Languages and Architectures for Image Processing", edited by M. Duff. and S. Levialdi [DUF81] a number of languages and systems for image processing are presented and compared [MAG81]. The current IOS compares favorably to all of the languages presented. The VISIONS IOS seems to integrate many of the strengths of the other image analysis systems. There are two important characteristics which distinguish the VISIONS IOS from the other image analysis systems:

(1) The partitioning of the image analysis problem into interpretive control and non-interpretive image operators allows for good dynamic control for interactive experimentation without sacrificing image operator efficiency.

(2) The structure imposed on the image operators simplifies the construction of these operators to minimize image operator development time and cost.

Future IOS Development.

Further development of the IOS should include the implementation of the history and menu subsystems. A subsystem to help users interactively program the prototype image operators is also being developed. Special faster image data base accessing functions are being developed to improve the efficiency of many of the shared image operators. Since the shared operators are used very frequently, these EDBAFs should enhance the IOS's efficiency considerably.

The major pending modification to the system will distribute the EDB in the virtual space. Image operators would be subprocesses which would communicate with the IOS through shared memory (the SDB and sections of the EDB actually used by the image operator would be shared). This would allow for multiple tasks, such as an image operator task and a display task, to execute concurrently. This means a user could initiate the display of a particular image and then initiate the computation of another image operator on a different image without having to wait for the display operator to complete. The primary advantage of the new system would not be derived from the multi-processing capability, but from the shortening of the debug cycle for image operators. In the new system, image operators could be linked independently without the massive LISP system, allowing the duration of the debug cycle to be reduced to about five percent of its duration in the current IOS (i.e. from minutes to seconds). It will be possible to test, edit, compile, link, and retest a new image operator without ever leaving the IOS.

3.0 BACKGROUND, MOTIVATION, AND CONTEXT

This chapter should provide all of the necessary background for the research contributions presented in Chapters 4, 6, and 7. The chapter is organized into four separate sections.

The first section provides some general background into the problem of image segmentation. Research into picture processing and scene analysis has been progressing for almost twenty years since Roberts described one of the first scene analysis systems in 1963 [ROB63]. Now hundreds of papers are published each year on segmentation and image analysis [ROS80b]. This section does not attempt to provide a comprehensive review of the field but instead provides some necessary pointers into the literature for the papers particularly relevant to this thesis.

The next section contains a discussion of the approach taken in this dissertation. This section contains a discussion of another system (Hearsay II) which also attempted to integrate knowledge from multiple sources to solve a single problem.

The third section reviews the few other attempts at integrating several segmentation algorithms. Chapter 7 addresses this problem in this dissertation.

The fourth and last section describes the particular segmentation algorithms which were utilized in the thesis. In particular, an iterative clustering algorithm [NAG79], an iterative edge finding algorithm ([HAN78a]), and a thresholding segmentation algorithm ([KOH79], [KOH81]) are described.

3.1 Characteristics of Segmentation Algorithms

Segmentation algorithms can be classified along a number of different dimensions. One of the most important is a distinction of segmentation algorithms which are based on edges from those based on regions. The region algorithms form regions explicitly based on some similarity measure which groups similar adjacent pixels into regions. Our goal in the VISIONS system is to label regions in the segmentation with some semantic identity. Given this goal, the edge algorithms may be viewed as implicitly forming regions by locating the discontinuities or region boundaries at which large differences between adjacent pixels occur. This difference in approach to the segmentation problem can result in dramatically different partitionings of the image. Many of the edge algorithms are reviewed in [HAN80] and [DAV75] while many region algorithms are reviewed in [NAG79], [OHL75], and [PRI77]. An important goal of the current thesis is the integration of these disparate approaches. Some progress has been made in this area and this work is discussed in section 3.4 below.

Segmentation algorithms may be based on local or global information for both edge and region algorithms. Global techniques are used in clustering [OHL75], [PRI77], [PRE66], [TSU73], [NAG79], or in threshold selection [KOH81], [KAT65], [WAT74] among others. However, global approaches may be based on assumptions which do not hold locally, while local approaches may jump to local conclusions which are incorrect in some more global context.

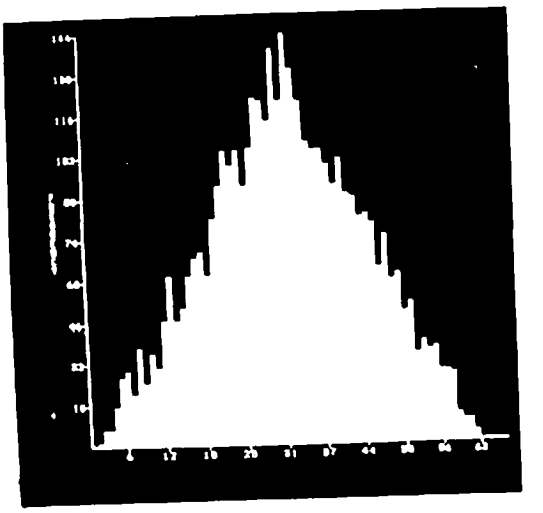
Some of the problems of global approaches based on feature histograms, including the problems of overlapping distributions and hidden clusters, are addressed in [NAG79]. The two techniques most commonly used to overcome the problems of the global approach are localization of the global approaches to arbitrary subimages ([CHO71], [NAG79]) and recursive application of the global segmentation approaches ([OHL75], [PRI77]). Below we provide two simple examples to illustrate these approaches.

Let us assume that we are going to segment images using an algorithm which selects clusters (or peaks) from a histogram of pixel intensities and then labels each pixel with the cluster label closest to the intensity value of the pixel (i.e. a minimum distance classifier in feature space). Figure 8 shows how localization in the image space can lead to more reliable cluster selection. Figure 8a shows an image with a small square on a background with a small but wide intensity gradient (i.e. slow spatial variation). Figure 8b shows the essentially unimodal histogram of this image. A segmentation based on this histogram would not be able to distinguish the square from the background since there is not a separate peak in the histogram corresponding to the small square. However, if we partition the image into 16 square subimages of 16 by 16 pixels and segment each subimage independently, then we can find separate clusters for the small square and the background. Figures 8c and 8d show the subimage containing the small square and the corresponding histogram. Clearly the cluster corresponding to the small square could be found in the localized histogram. Localization is especially useful in images with small objects but it should be noted that the set of local segmentations produced must be reconnected into a single global segmentation (a non-trivial problem).

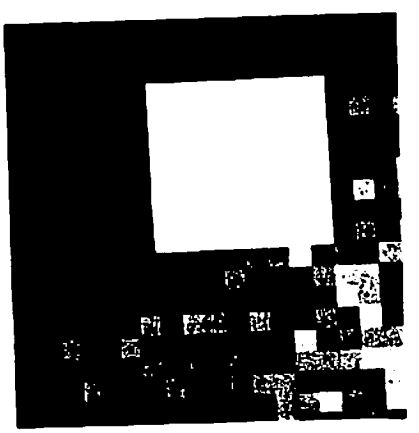
Ohlander, Price, and others used recursive histogramming instead of localization to capture clusters obscured in the global feature histogram. In this method a cluster is selected and all image pixels are labeled to be in the cluster or not, thus forming regions in the image. For each large region a histogram is computed, and if the histogram is not unimodal, a cluster is selected and the process is recursively applied until all regions are small or unimodal. Figure 9 shows how recursive histogramming can lead to more complete segmentations. Figure 9a shows an image to be segmented. The figure consists of three rectangles on a dark background. The two leftmost and adjacent rectangles have quite different intensities (μ=30 and μ=40) while the right rectangle has an intermediate intensity (μ=35). Figure 9b shows the histogram of the whole image. The first histogram allows for the separation of the foreground rectangles and the background. Figure 9c shows the segmentation produced by the global histogram alone. This segmentation has not separated the two adjacent rectangles on the left. The recursive segmentation approach would now take all large regions of this segmentation and apply the same segmentation algorithm to each region. The histogram of the regions corresponding to the background and the right rectangle are essentially unimodal and therefore are not segmented further. However, the region corresponding to the two left rectangles (shown in figure 9d) is bimodal as seen in figure 9e. Figure 9f shows the final segmentation produced by the intersection of the segmentation of the global segmentation and the partition of the left region based on the histogram of figure 9d.

Region algorithms based on global clustering are often subject to gross errors when clusters are hidden or when distributions overlap in the feature space (histogram). The use of recursive clustering is often effective, but is particularly

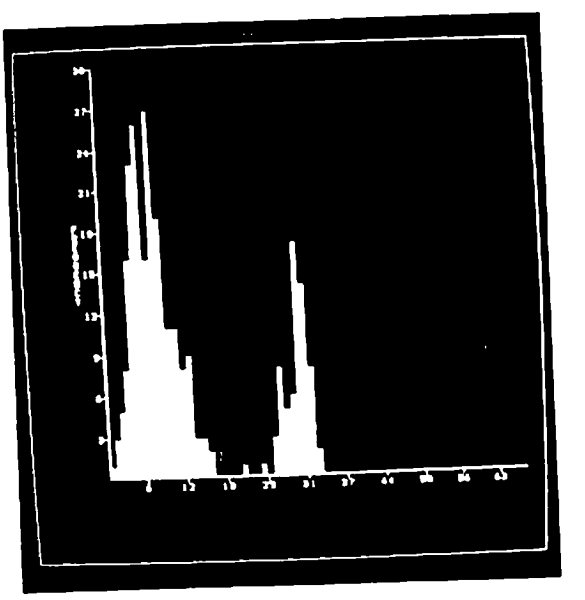
(a)



(b)

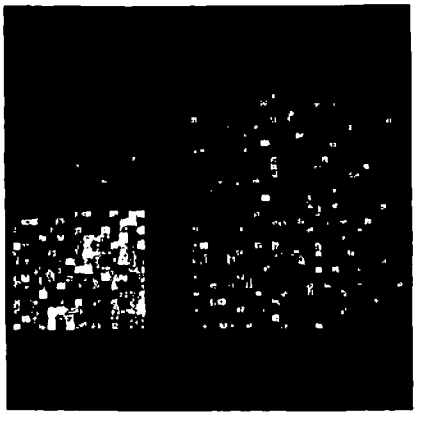


(c)



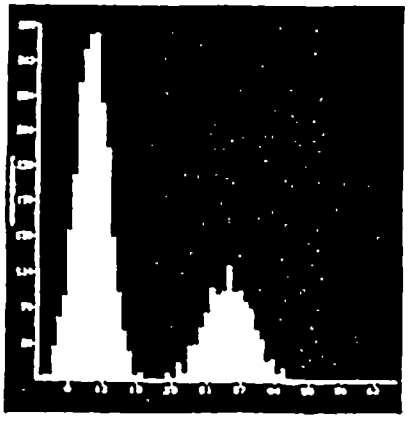
(d)

Figure 8: Using Localization in Cluster Based Segmentation.

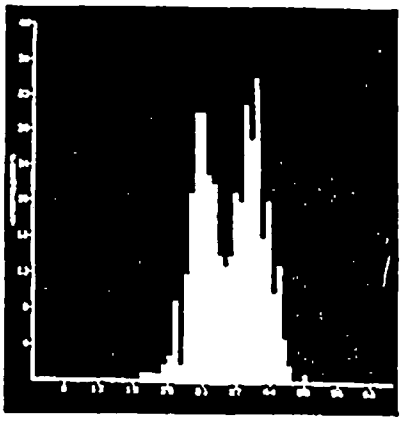


(a)

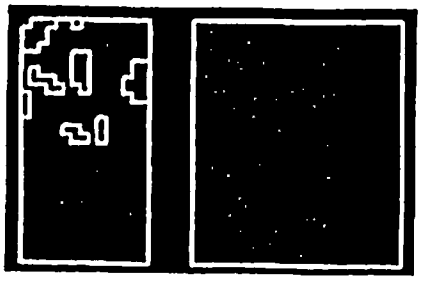
(d)



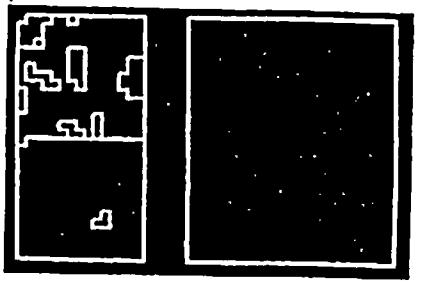
(b)



(e)



(c)



(f)

Figure 9: Using Recursion in Cluster Based Segmentation.

sensitive to the size of the objects to be detected. In figure 8a the small square could not have been separated using recursive clustering since the global histogram is effectively unimodal. Localization is less sensitive to the size of the object and can successfully segment the square in figure 8a. Localization is the method used to overcome the problems of global clustering in the remainder of the thesis.

Edge algorithms are generally considered to be local when the data upon which they operate is derived from some local differencing operator; however, the algorithms for producing a segmentation based on edges may have both local and global components. For instance, the threshold for edge presence in particular edge algorithms may be defined either locally or globally [HAN78a]. There are global histogram algorithms which detect lines using Hough transforms of edges [HOU62]. These algorithms perform global clustering of edge location and orientation to locate collinear segments. The edge algorithm used in the remainder of this dissertation computes edges using local contrast measures.

Another characteristic of segmentation algorithms is the generality of the algorithm or the range of image classes to which the algorithms may be effectively applied. Many algorithms are designed to operate in restricted domains, such as particular types of biomedical images ([PRE66], [WES75]), or polyhedral blocks in an environment with controlled lighting ([ROB63], [GUZ68], [WAL75]). Segmentation algorithms can be quite effective in these constrained domains since the algorithms may be selected and carefully tuned by taking advantage of the known characteristics of the images. Some investigators have used top-down prediction based on semantic models of the scenes to facilitate the segmentation process ([YAK73], [BAL77], [TEN76]). Semantic information has been used directly for image segmentation where areas of the image are labeled with potential object labels and corresponding probabilities [YAK73]. It is doubtful that such schemes for integrating semantic knowledge directly into the segmentation phase will succeed with such simple single level approaches when the number of objects in the image domain becomes very large.

While the use of semantic information to guide the segmentation process has potential in specialized applications, it is our position that segmentation algorithms should probably not be based on knowledge of the objects to be recognized. The view taken here is that higher level processes, which utilize semantic knowledge, should influence or tune the low level segmentation processes, but semantic knowledge such as known object identities should not be utilized directly by the segmentation processes. This will, hopefully, result in a domain independent segmentation system which can easily be tuned to take advantage of prior knowledge for a given domain.

3.2 Segmentation Executive Components

In Chapter 2 it was shown that simple image operators could be easily implemented in the Image Operating System. These operators can perform simple calculations in parallel for each pixel, producing result planes at the same level of resolution (iteration), at higher levels of resolution (projection), or at lower levels of resolution (reduction). These operators alone do not constitute a segmentation algorithm. Rather, we define a **segmentation algorithm** as a sequence of such operators which can accept an image and a set of parameters as input and produce a segmentation as output.

Segmentation processes are defined as instantiated segmentation algorithms. A segmentation process can be executed only after the selection, via some strategy, of a segmentation algorithm, appropriate image features which determine the input data (e.g. raw red, saturation, intensity, etc.) and appropriate algorithm-specific parameters.

A complete segmentation computation could consist of a composition of several segmentation processes, with pre-processing and/or post-processing algorithms. One preprocessing algorithm used in our research attempts to correct for noise due to digitization [OVE79], while another uses color information to correct boundary blurring [PRA80]. One post-processing algorithm suppresses very small regions, while another (an iterative expand and contract algorithm) closes small gaps in an edge segmentation in order to form closed boundaries [PER80].

In our effort to develop intelligent and effective segmentation strategies, it will prove helpful to characterize the constituent image operators according to their function. Some of the image operators introduce new hypotheses into the representation based on the image data; these include operators which generate hypotheses about cluster membership, or the probability of the existence of an edge between each pair of pixels. There are other image operators which update segmentation hypotheses based on local constraints, e.g. the relaxation operator for edges attempts to organize local edges into continuous global boundaries.

There are also image operators which abstract several hypotheses at some level of resolution into a single more global hypothesis at a coarser level of resolution. Other image operators project abstracted hypotheses to hypotheses at higher (finer) levels of resolution. For instance, an operation might aggregate high resolution edges into a straight boundary hypothesis at low resolution. The low resolution boundary hypothesis could then generate edge hypotheses or increase the confidence of existing edge hypotheses for the implied constituent edges at the higher level of resolution in the cone. It is the above characterization of image operators that leads us to view our segmentation system in terms of other systems which solve complex problems by a hypothesis and test paradigm involving multiple knowledge sources such as Hearsay-II.

3.3 The Hearsay-II Analogy

The task we have defined for a segmentation executive includes the construction and control of the hierarchy of processing structures described above. A certain similarity exists between the task of the segmentation executive and other systems designed to coordinate multiple, independent, cooperating knowledge sources such as Hearsay-II [ERM80] in speech understanding, or the VISIONS scene interpretation systems in image understanding ([HAN74], [HAN78b]).

The Hearsay model had a large number of independent knowledge sources which communicated through a global common data structure called a blackboard. Each knowledge source had a limited view of the world state represented in the blackboard and each knowledge source was considered to be both "incomplete" and "errorful". No single knowledge source could solve the entire problem alone, and even the partial results produced by a knowledge source might have been incorrect. It was assumed that other knowledge sources would supply the missing information, while the incorrect hypotheses would be corrected or ignored during the interpretation as evidence accumulated.

The knowledge sources were data invoked and were scheduled when a particular data event in the blackboard occurred. The scheduler focussed the attention of the system by ordering operations on the scheduling queue according to an evaluation based on the expected effect of performing the operation. The scheduler would ignore operations in portions of the utterance which were well-understood since little new information could be gained by the application of these operations. The scheduler would normally choose to concentrate on areas adjacent to unambiguous portions of the utterance since these well-understood sections formed "islands of reliability" which could greatly constrain the search. In Hearsay-II it was found that when hypotheses at the lower levels of understanding (i.e. phonemes) improved even slightly, the overall performance of the system improved markedly; the search space was considerably reduced and the correct paths were evaluated earlier in the search.

This last finding implies that it is important to produce the best possible segmentations to enhance the performance of the semantic interpretation system. The primary influence of Hearsay-II and other such systems on this thesis does not arise from their system architecture, but from the overall approach to the problem of deciding between alternative competing hypotheses produced by incomplete and error-prone knowledge sources. The parallel nature of the low-level image processing architecture makes focus of attention in the image space less important, since the image operators are applied in parallel to the entire image.[8] In our

[8] This assumes that parallel hardware exists to execute the image operators. Spatial focus of attention is of considerable concern in a sequential implementation since it has a tremendous impact on processing time and cost. The cone hierarchy can be used to implement coarse resolution plans to limit processing to some portion of the image.

implementation of the segmentation executive, the focus of attention mechanisms decide which segmentation processes (or KS's) to apply and when. The high processing cost of performing the segmentation process justifies considerable cost in making scheduling or KS selection decisions.

How do KS's in Hearsay-II relate to our segmentation processes? One major difference is in the size of the KS's. The segmentation processes are tightly coupled sets of image operators which generate new hypotheses throughout the image while Hearsay's KS's are small, relatively efficient, and generate a very few hypotheses which are usually restricted to a small portion of the utterance. As in Hearsay-II, interactions between algorithms must take place via a common interface. In Hearsay this interface was a hierarchical structure containing hypotheses known as a blackboard. For segmentation processes the processing cone serves this function. Unlike Hearsay, image operators are not data-invoked (invoked automatically by the generation of new hypotheses or the modification of old hypotheses).

3.4 Attempts at Reconciling Edges and Regions

One of the expressed goals of this thesis is to integrate the knowledge encoded in several algorithms or segmentation processes to generate better segmentations. There are several possible approaches to the integration of the region and edge algorithms. This section briefly reviews other work which implicitly or explicitly attempts to combine both region and edge information.

One interesting approach is to define appropriate models of image formation based on both region and edge information. The slope facet model proposed by Haralick models the image as a large number of small parameterized facets (bilinear, quadric, or higher order surface patches of the intensity or other feature surface). Edges are found when adjacent facets have significantly different parameters, while regions are aggregated by grouping facets with similar parameters. Region and edge detection are based on standard statistical measures (analysis of variance) [HAR79].

This dissertation attempts to integrate edge and region information by integrating edge information into the region algorithm, by integrating segmentations produced by the independently computed region and edge based algorithms, and by dynamically integrating the region and edge algorithms.

The building of segmentation processes includes the selection of algorithm specific parameters. Selection of these parameters should consider region information for edge algorithms and edge information for region algorithms. A number of region algorithms based on thresholding the image have utilized gradient or edge information (including [WES78], [KAT65], [WAT74], [MIL78], [MIL79], and [KOH81]). Kohler [KOH81] contains a comparative evaluation of these methods.

Some very preliminary experiments with dynamic interaction of segmentation processes have been reported by Webster [WEB79]. This work independently proposed a model of process interaction which is strikingly similar to the model proposed in this thesis (in chapter 7 below). Webster closely coupled a region relaxation algorithm and an edge relaxation algorithm.[9] Inter-pixel edge probabilities were substituted for compatibility coefficients, thus allowing the edge algorithm to influence the region hypotheses. Region label hypotheses were designed to influence the edge algorithm by averaging the probability that adjacent regions are labeled differently into the edge probability between the pixels.

The results reported by Webster were not encouraging since no significant improvements in the segmentation due to process ,interaction were found. The failure to find improvement may have been due to a number of factors. The algorithm was tested on a single test scene (a small square on a uniform background) with different levels of additive noise. This kind of image does not exhibit any of the complex image characteristics (such as intensity gradients, non-gaussian texture, and fine image structures) which are difficult for region based algorithms to deal with. This implied that the edge hypotheses could provide little help to the more reliable region hypotheses and, therefore, the quality of the resulting segmentation would not be improved. The lack of improvement may also have been due to a failure to tune the edge algorithm independently and a failure to consider the relative contribution of the region and edge components during the interaction process.

3.5 The Region Algorithm

The cluster-based region algorithm used in the remainder of this dissertation is a variation of a relaxation algorithm developed by Nagin [NAG79]; extensions to this algorithm are discussed in chapter 4. This algorithm is based on the assumption that the regions in an image will form distinct clusters in feature space (a histogram of the feature values). Regions are formed by identifying clusters in feature space and ultimately labeling each of the pixels with one of the cluster labels. Adjacent pixels will be aggregated into the same region if they have identical labels, i.e. they belong to the same cluster.

The algorithm has two phases. In the first phase cluster centers are located in the feature space and for each pixel a probability of cluster membership for each of the possible clusters is computed. The clusters selected will be referenced by the label set L={i | 1<=i<=m} where m is the number of clusters detected. In this thesis, only one dimensional histograms are used for finding feature clusters, although it is possible to look for feature clusters in higher-dimensional feature spaces (in particular [NAG79] uses 2 dimensional histograms and [COL78] uses unsupervised n-dimensional clustering). The initial label probabilities for cluster

[9] The region relaxation system is very similar to [NAG79] and the edge system is very similar to [HAN80].

affiliation at a given pixel were determined in a straightforward manner using the normalized inverse distance of the pixel feature value to each of the selected cluster centers.

The second phase of the algorithm is a probabilistic relaxation process [ROS76] which updates the cluster label probabilities at each pixel based on a five-neighbor context (the four adjacent pixels and the center pixel itself). The neighborhood will be referred to as the set N. The contribution of each neighbor $x \in N$ to each cluster label i at the center pixel is defined to be:

$$\sum_{j \in L} r_x(i,j) * P_x(j) \qquad 3.1$$

where $P_x(j)$ is the current probability that neighbor x is correctly labeled j, and $r_x(i,j)$ is the compatibility coefficient which reflects the support for label i at the center pixel given that neighbor x is correctly labeled j. The compatibility coefficient is based on the compatibility between label i at the center pixel with label j at neighbor x in the initial cluster label distribution. The relative support for each of the labels at the center pixel is obtained by summing over all neighbors. The support for label i at the center pixel is defined to be:

$$q(i) = \sum_{x \in n} \sum_{j \in L} r_x(i,j) * P_x(j) \qquad 3.2$$

Normalization of these contributions via the standard relaxation formula from Rosenfeld [ROS76] allows the computation of a new probability for label i at the central pixel ($P_c(i)$) as follows:

$$P_c(i) = \frac{P_c(i) * (1 + q(i))}{\sum_{j \in L} P_c(j) * (1+ q(j))} \qquad 3.3$$

The relaxation process is typically run 20 to 50 iterations and then each pixel is assigned the highest probability cluster label. Note that there have been a variety of updating formulas proposed and there are a variety of approaches for specifying the compatibility coefficients. In [NAG79] the compatibilities are estimated from the original probability distribution of cluster labels and do not change during the relaxation phase. It has been shown that in some instances these image-dependent distributions capture important contextual information that allows fine image structures to be preserved within the segmentation.

Nagin used localization to avoid missing clusters. The image is divided into small subimages (typically 16x16 or 32x32 pixel subimages) and the algorithm is independently applied to all subimages. The histogram for cluster selection is

computed over a subimage which extends 25% in each direction beyond the subimage in which the segmentation is computed. This is done in order to reduce the risk of missing clusters corresponding to objects which straddle the inter-subimage boundaries. The result of this processing is a set of segmentations, one for each subimage, which must be combined into a single segmentation. Nagin integrated these segmentations by merging regions across subimage boundaries. The criterion for merging was based on a measure of the difference between the populations of pixels belonging to the candidate regions in a narrow band along the sub-image boundary. Because the merge decision is local, it is possible to indirectly link two very different regions into the same region via a chain of regions, each of which is only slightly different from its neighbors in the chain. This is particularly noticeable when the two very different regions being merged happen to lie in the same subimage. These merge errors are often due to missing clusters in some subimage, a problem addressed in chapter 6.

3.6 The Edge Algorithm

The edge algorithm used in the remainder of this thesis was developed by Hanson, Riseman, and Glazer [HAN80] (this work evolved from a similar algorithm by Prager [PRA79]). No modifications have been made to this algorithm and except as discussed in the section on dynamic segmentation process interaction below, the algorithm utilized was identical to that presented in [HAN80].

The edge algorithm is an iterative algorithm which attempts to organize local edges into continuous line segments which correspond to region boundaries. The edge algorithm consists of a sequence of image operators which generates a set of local edge hypotheses, followed by a relaxation sequence which modifies the edge hypotheses based on constraints in a local context around the edge hypothesis. In this algorithm edges are considered to exist between pixels as suggested by [HAN78b] and [PRA79].

To generate the initial edge hypotheses the algorithm first uses two simple one-by-two edge masks to measure both the horizontal and vertical local edge contrast at each pixel. These initial edge hypotheses, based solely on absolute local contrast, are then locally scaled by an image operator which normalizes the edge contrast based on a function of the highest local contrast in an 11 by 11 neighborhood of the edge. The initial edge hypotheses are also adjusted by an image operator which collects a set of parallel non-zero gradient edges of the same direction of contrast into a single more global boundary. This "gradient collection" process overcomes some of the problems of using the very local one-by-two edge mask. At this point, initial edge hypotheses are represented as a probability that an edge should be present in the final edge segmentation at that location.

The second phase of the algorithm is an iterative relaxation process across the plane of edge hypotheses. Using assumptions about good line continuation, the probability of each edge is updated on the basis of possible boundary continuation

on either side of the edge. Figure 10 shows the ten equivalence classes V_{ij} of edge contexts considered by the relaxation operator, where i and j represent the number of edges present to either side of the central edge. In a given edge context we estimate the probabilities that this edge context is an instantiation of each of these equivalence classes. In the V_{00} case the central edge has no support from any of its neighbors, while in the V_{02} and V_{03} cases the central edge is not needed for good continuation and to the extent that we believe the current edge context to be an example of one of these cases the probability of the central edge is decreased toward zero. If we believe an edge context to be an example of class V_{01}, then the edge is one of three possible continuations of the extant neighboring edge, one of which should exist. In this case the probability of the central edge is increased slightly. If we believe the edge context to be an example of the classes V_{11}, V_{12}, or V_{13}, then the central edge is necessary for good continuation and the probability of the central edge is increased toward one. The remaining cases are ambiguous in terms of what is required for good continuation and have no effect on the probability of the central edge. Additional information used in the updating includes the consistency of the edge's direction (i.e.. the signs of the gradients) and the alternative parallel locations for placement on either side of the edge.

The edge relaxation is typically terminated after about 20 iterations. Figure 11 shows a typical result of this algorithm on a natural outdoor scene. Figure 11a shows the original intensity image, Figure 11b shows the initial edge probabilities, and Figure 11c shows the resulting edges after 2 iterations of relaxation and figure 11d shows the result after 20 iterations. Note that although many of the discontinuities in the image have been properly located, the segmentation often does not form closed regions which correspond to real world structures. Some additional processing is clearly needed before this segmentation can be properly utilized by a semantic interpretation system which requires closed regions.

3.7 The Thresholding Segmentation Algorithm

Another algorithm used later in this dissertation is a multi-threshold segmentation algorithm [KOH78], [KOH81]. The algorithm selects a threshold for a given image such that the average intensity gradient across all boundaries detected by the threshold is maximized. This selection is accomplished without search by simultaneously computing the expected average contrast (gradient) for each possible threshold. Additional thresholds can be identically selected after any edges detected by previously selected thresholds have been eliminated from the computation.

This algorithm, like the region algorithm, is guaranteed to form closed boundaries and is based on global measurements across the image. It is like the edge algorithm in that the threshold selection is based on local contrast information rather than similarity information as the region algorithm is.

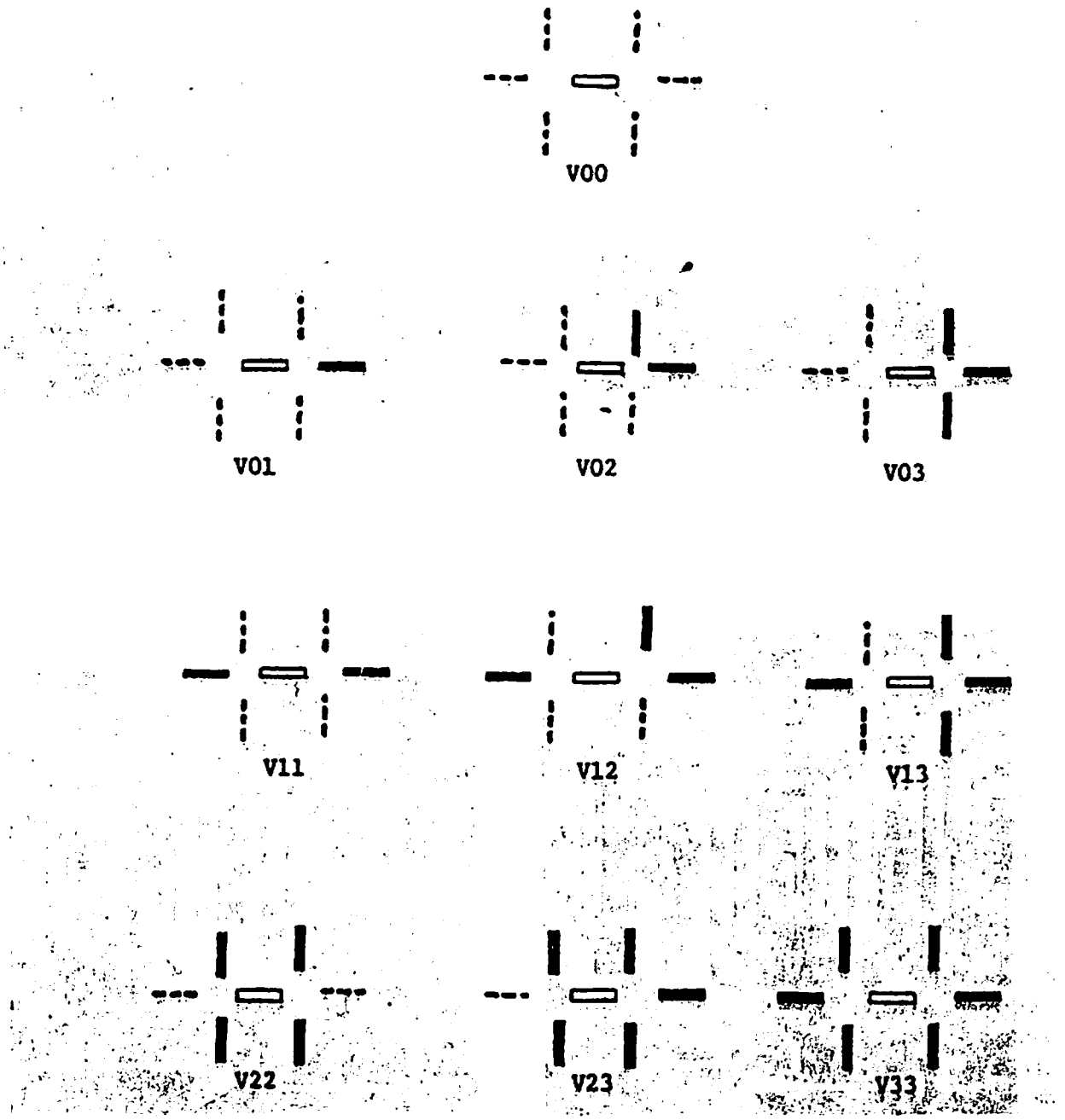
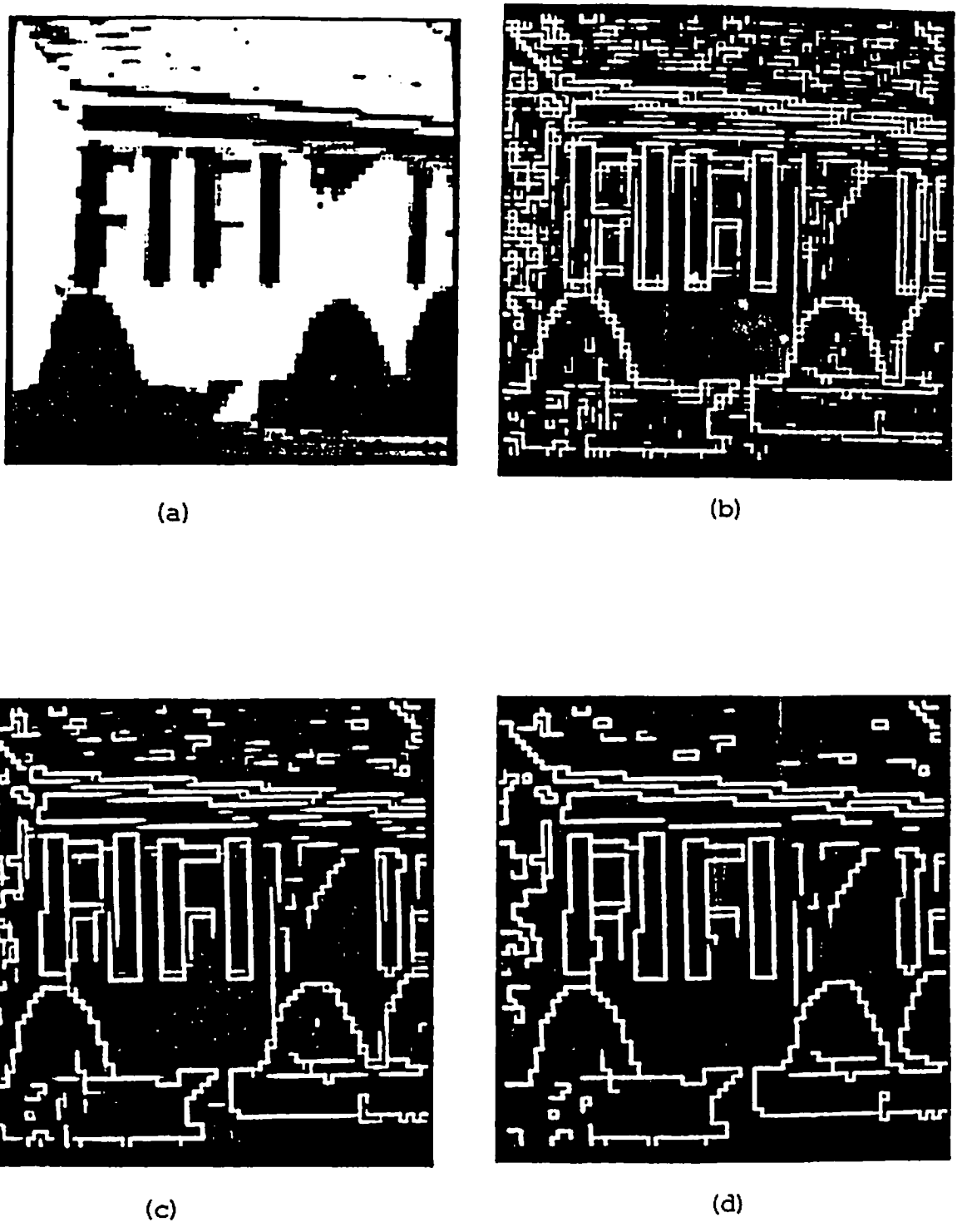


Figure 10: Equivalence Classes for Edge Contexts in the Edge Relaxation Algorithm.



(a) (b)

(c) (d)

Figure 11: Edge Relaxation Algorithm Segmentation Result.

4.0 MODIFICATIONS TO THE CLUSTERING ALGORITHM

Two modifications have been made to improve the performance of the Nagin cluster based segmentation algorithm. These are necessary and significant improvements to the region clustering algorithm which forms the primary segmentation process around which chapter six of this dissertation is centered. The first modification altered the computation of the initial cluster affiliation probabilities to reduce the classification error rate after relaxation. The second modification altered the definition of the compatibility coefficients such that the cluster label probabilities do not diverge in a manner dependent on the absolute magnitude of the compatibility coefficients.

4.1 Modification of Initial Probability Computation

The region algorithm selects feature space clusters and then associates a probability vector with each pixel. The component elements of the probability vector, which represent the likelihood that the pixel is a member of the respective clusters, are inversely proportional to the pixel's normalized distance to each of the cluster centers in the feature space. Given n cluster centers C_1 to C_n and a pixel with feature value i (e.g. intensity = i) then the probability component for cluster m would be given by:

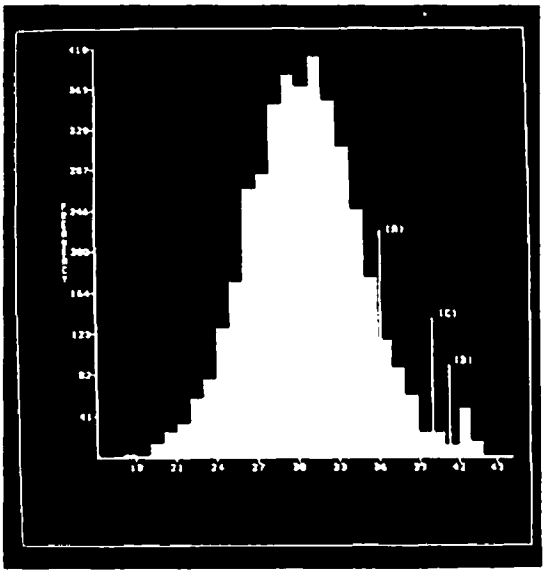
$$P(m) = \frac{1 / D_{im}}{\sum_{1<a<n} 1 / D_{ia}} \qquad 4.1$$

where D_{ia} is the feature space distance (e.g. intensity difference) between cluster center C_a and the feature value i. The values of the probability vectors are modified by an iterative updating process and the best label for a pixel is defined to be the most likely cluster label after relaxation. Note that a maximum likelihood classification could be carried out on the initial probability vectors, which is equivalent to a minimum distance classifier, since the maximum component of the probability vector always corresponds to the closest cluster center.

It is well known in pattern recognition [DUD73] that a minimum distance classifier is not always optimal. Since we have no knowledge of either the number of underlying distributions in the histogram or their form, a theoretical analysis seems intractable. However, another heuristic decision boundary at the valleys between the cluster peaks has been widely used [PRE66],[OHL75]. Figure 12a shows a simple image composed of two gaussian regions of different sizes and feature distributions (background $\mu = 30$, $\sigma = 4.0$, and foreground $\mu = 42$, $\sigma = .6$). Figure 12b shows the feature histogram with the minimum distance decision boundary (at a), the valley decision boundaries (at b)[10] while figures 12c and 12d

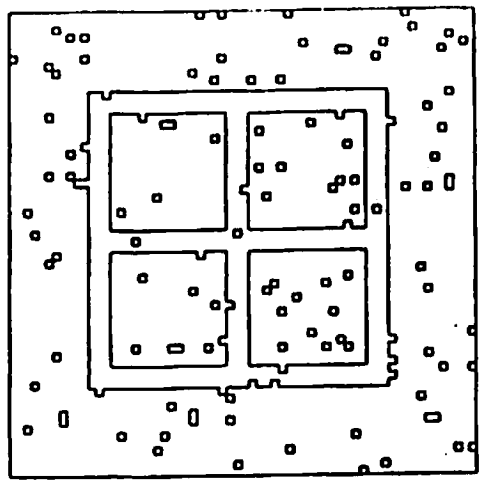
[10] Figure 12b also shows the heuristic decision boundary (at c) presented below.

(a)

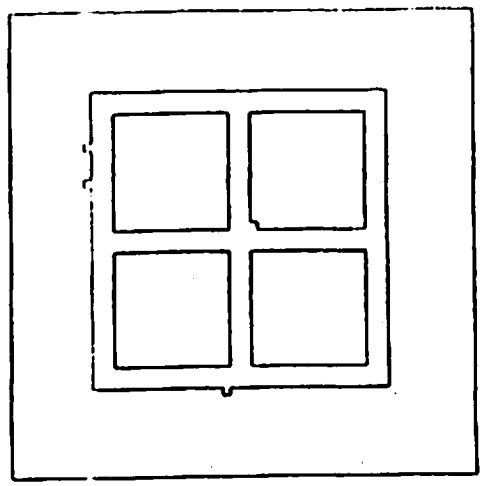


(b)

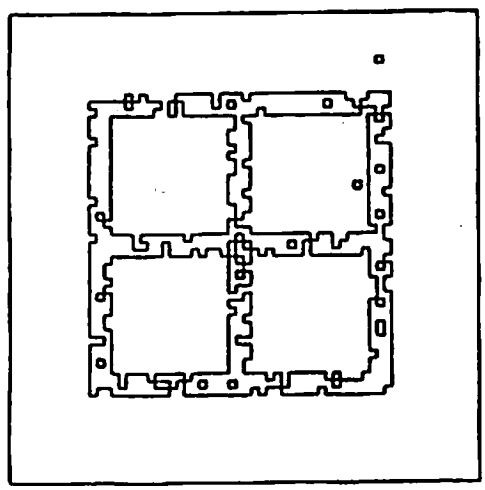
Figure 12: Decision Boundary Selection: Example 1.



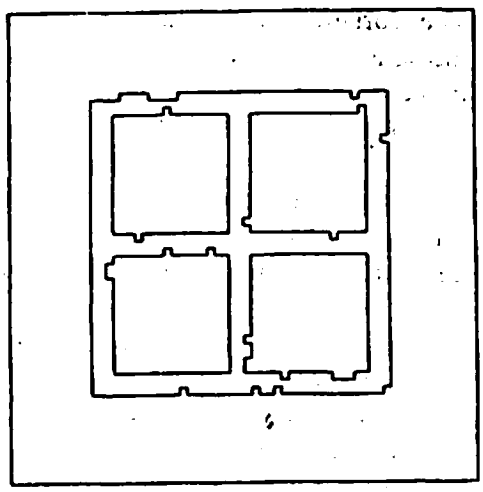
(c)



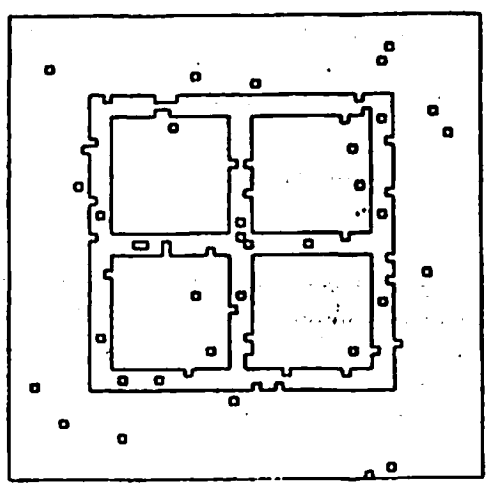
(f)



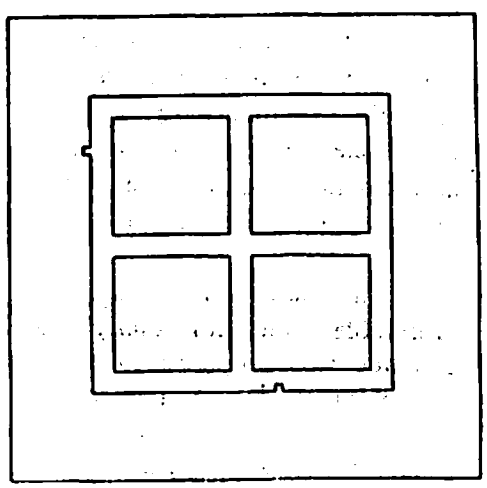
(d)



(g)



(e)



(h)

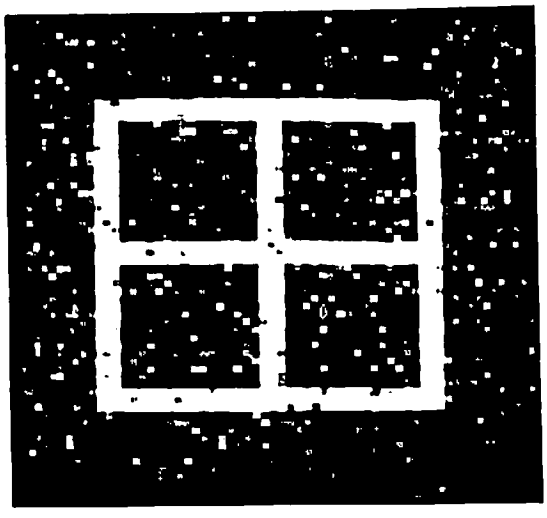
Figure 12: Continued.

show the initial segmentation based on the minimum distance and valley decision boundaries respectively. Use of the valley decision boundary has reduced the number of misclassified pixels from 221 to 16. This strongly suggests the use of the valley decision boundary if one assumes all errors have equal cost. However, in the context of the relaxation algorithm, all errors do not have equal cost since the probability updating may correct some errors. Figures 12f and 12g show that after 30 iterations of relaxation the minimum distance classifier (used by Nagin) resulted in 9 mis-classified pixels while the valley decision boundary resulted in an almost entirely correct segmentation. In this example, the valley decision boundary results in fewer errors both before and after the relaxation process.

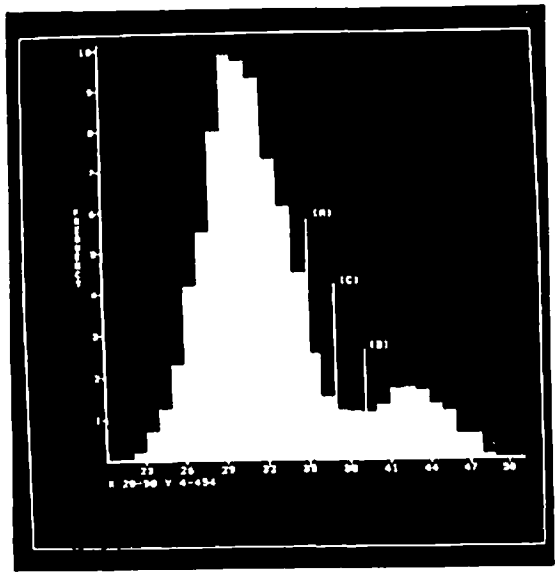
There are cases for which the valley classifier is inferior to the minimum distance classifier after relaxation, as shown in figure 13. Figure 13a shows a foreground region ($\mu = 42$, $\sigma = 3.0$) on a darker background ($\mu = 30$, $\sigma = 3.0$). The foreground region has a large perimeter to area ratio. Figure 13b shows the histogram of figure 13a with the minimum distance and valley decision boundaries marked at 'a' and 'b' respectively (the heuristic decision boundary proposed below is shown at 'c'). Figures 13c and 13d show the initial labeling of the pixels using the minimum distance and valley decision boundaries respectively. Figures 13f and 13g show the corresponding labeling after 30 iterations of relaxation. In these cases the minimum distance classification led to 12 errors after relaxation, while the valley decision boundary classification led to 41 errors.

The better performance of the minimum distance decision boundary in this last example seems to be due to two factors: (a) the significant difference in the sizes of the regions and (b) the large ratio of perimeter to area for the smaller region. Spatially adjacent errors in the interior of a region may not be corrected by the relaxation since they mutually support each other's incorrect label. Likewise, errors on the boundary of a region may have support for their incorrect label from the neighboring region. In this example, the selection of the valley decision boundary shifted the decision boundary toward the smaller cluster with a relatively high perimeter to area ratio, and therefore increased the number of errors which were difficult to correct in the small region. The probability of spatially co-occurring errors clearly increases as the same number of errors are squeezed into a smaller and smaller region. The probability of errors on the boundary of the region also increases with the perimeter to area ratio.

In cases such as the one above, the minimum distance decision boundary is preferable, yet, for many cases the valley decision boundary is preferable. A heuristic was proposed to move the decision boundary away from the smaller of the two populations to reduce the probability of spatially co-occuring errors in the smaller region. The heuristic was designed to move the decision boundary from the valley boundary toward the minimum distance boundary by an amount proportional to the difference in the sizes of the populations and the degree to which the populations overlapped. The heuristic decision boundary, simply based on the histogram, was defined to be:



(a)



(b)

Figure 13: Decision Boundary Selection: Example 2.

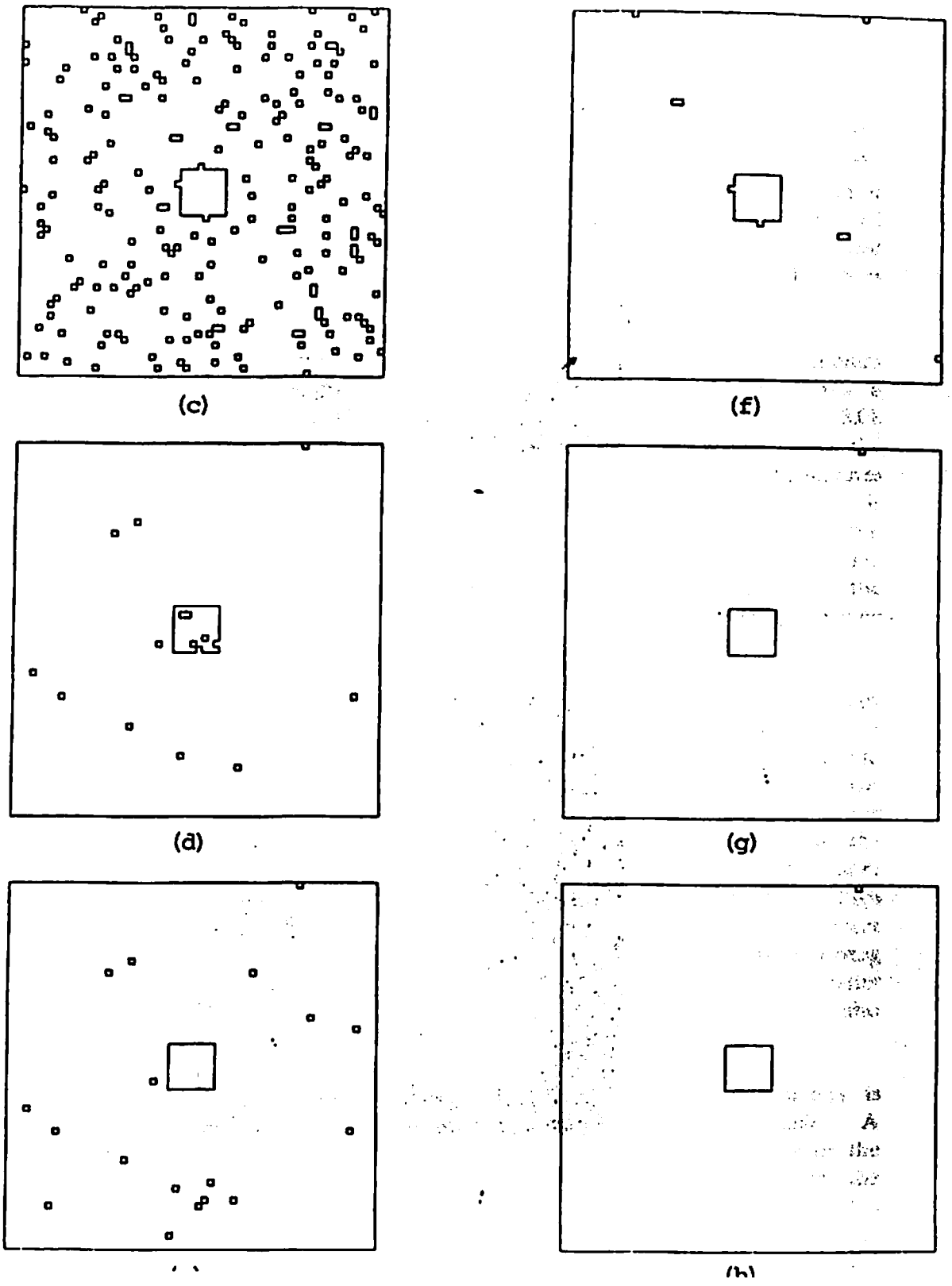


Figure 13: Continued.

$$D = \frac{V}{\min(H1,H2)} * \left(1 - \frac{\min(H1,H2)}{\max(H1,H2)} \right) * (D_{md} - D_{vd}) + D_{vd} \qquad 4.2$$

where D_{md} and D_{vd} are the minimum distance and valley decision boundaries respectively, V is the feature frequency found at D_{vd}, while H1 and H2 are the feature frequencies at the cluster centers. The V term is a crude estimate of the population overlap, while H1 and H2 are crude estimates of the population sizes. The resulting segmentations before and after relaxation are shown in figures 12e and 12h and 13e and 13h for the two example images.

In the case of figure 12 the heuristic decision bound is very close to the valley decision boundary (at i=40 rather than i=41) and results in one error after relaxation. In the case of figure 13 the heuristic decision boundary was placed halfway between the valley and minimum distance decision boundaries (at i=37 where valley decision boundary selected i=39 and the minimum distance boundary was i=35). In the second case, the minimum distance boundary resulted in 12 errors after relaxation, the valley boundary resulted in 41 errors, and the heuristic boundary resulted in only 3 errors.

4.2 Modification of Center Pixel Compatibilities

In the Nagin relaxation algorithm each neighbor of a pixel to be updated has a distinct set of compatibility coefficients which define how that neighbor influences the probability update at the central pixel. All of these compatibility coefficients are determined from the initial probability distribution of the image based on a correlation measure. However, the choice of compatibility coefficient for self support of the central pixel in Nagin's algorithm was somewhat ad hoc:

$$r_c = \begin{cases} 1 & \text{if } i=j \\ -1 & \text{otherwise} \end{cases} \qquad 4.4$$

This section defines objectives for the relaxation in terms of the algorithm's behavior in particular image geometries at convergence or partial convergence, and then determines the appropriate central pixel compatibilities which will attain these objectives. This methodology provides understanding into the behavior of the relaxation algorithm in certain partially converged cases and therefore permits precise control of the relaxation algorithm in these cases. The analysis of the central pixel compatibility coefficients was suggested by the method of analysis used by Richards, Landgrebe, Swain [RIC80] in the determination of appropriate neighbor weights for each pixel in a different relaxation algorithm.

The Nagin center pixel compatibilities lead to considerably different behavior of the relaxation algorithm when the initial probabilities were very ambiguous or very clear. Empirically, when the initial probabilities were very close to 0 or 1,

then the magnitudes of the compatibility coefficients were large. In this case, the four neighbors overpowered the self support of the central pixel and the relaxation tended to destroy fine structures. If the initial probabilities of the best and second best labels were roughly the same for most of the pixels, then the magnitudes of the compatibility coefficients were quite small and the relaxation update was dominated by the central pixel. In this case, even isolated mislabeled pixels could survive the relaxation process.

Thus, one of the reasons Nagin's method did not always perform effectively is because it did not properly balance the influence of the central pixel with that of the other neighbors. The analysis below makes the assumption that certain geometries (such as long thin regions) are to be preserved by the relaxation process. This assumption gives rise to a set of constraint inequalities which may be solved for the values of the center pixel compatibility coefficients such that when local convergence is reached (i.e. probabilities for cluster labels are 0 or 1 in some image neighborhood) the desired geometry of labels remains stable.

In this analysis, we define two constraints on the relaxation process:

(1) The stability constraint.
A pixel which has converged to label x will remain converged to label x when at least one of its neighbors is also converged to label x. A pixel is converged if the probability of its best label is 1.

(2) The effectiveness constraint.
A pixel should not be allowed to remain converged to label x if all of its neighbors are converged and none are labeled x.

The stability constraint guarantees that one pixel wide, fine structures will be preserved and that two pixel regions are possible in the final segmentation. If two supporting neighbors were required for stability, then it would not be possible for one pixel wide fine structures to have the probabilities of their labels guaranteed to be stable since they might then erode from the end of the structure. When there are different geometries of labels whose stability is desired, a similar derivation is possible, although there is no guarantee that a solution will exist for other sets of constraints.

The effectiveness constraint guarantees that no isolated pixels will survive the relaxation process at convergence.[11] This condition guarantees that every converged pixel is part of a stable structure. Several alternative, less restrictive, definitions of the effectiveness constraint are also reasonable. It may not always be possible or desirable to have such a strong effectiveness constraint. The three other possible effectiveness constraints would guarantee that an isolated pixel would not remain at convergence only if at least either 2,3, or all four neighbors have converged to the

[11] By isolated we mean none of the pixel's neighbors are converged to the same label as the pixel.

same label. Note that all of these constraints are identical in the two label case. The theory does not guarantee that a pixel will converge to the same label as one of its neighbors, only that the pixel will not converge to a label different from every neighbor; however in practice, this is almost always the case.

For both the effectiveness and stability constraints, the particular constraint selected might be a function of the image domain, but the analysis presented below could be similarly derived for any of the possible constraints.

If the neighborhood is converged and the central pixel is labeled i, then that label will be maintained if the relaxation update equation (3.3) does not decrease the value of $P_c(i)$. From equation 3.3 we can see that this condition holds whenever

$$\frac{1 + q(i)}{\sum_{j \in L} P_c(i) * (1+ q(j))} >= 1 \qquad 4.5$$

Since the sum all $P_c(j)$ for all j in L is 1.0, the denominator of this inequality is a weighted average of the (1 + q(j)) terms. This means that the label i will be preserved if and only if q(i) is greater than the weighted average value of all q(j) for j in L. This is certainly true when q(i) >= q(j) for all other labels j (that is when the current label has more support than any other alternative label). Let us use $P_x(y)$ to denote the probability that pixel x should be labeled y.[12] Figure 14a shows the neighborhood used in the update equation for the case of two clusters. The following section shows how the center pixel compatibilities may be found to satisfy the stability constraint in the case of two labels. Section 4.2.2 extends the argument to multiple labels and section 4.2.3 derives the effectiveness constraint for these cases. Finally, section 4.2.4 extends the method to certain partially converged neighborhoods. Section 4.2.5 compares the region relaxation algorithm using the new center compatibilities to the Nagin algorithm in a simple experiment.

Stability Constraint - The Two Label Case.

We will initially restrict ourselves to the case of two labels in the neighborhood. If we assume that the update neighborhood is locally converged (probabilities of all labels are 0 or 1), then q(j) for each label j can be simplified since $P_x(y)=1$ or $P_x(y)=0$. For the case shown in figure 14b the relaxation update equation terms for q(1) and q(2) can be expanded as follows:

[12] The notation used for the neighbors is as follows:
c - center pixel
r - pixel to right
l - pixel to left
u - pixel above (up)
d - pixel below (down)

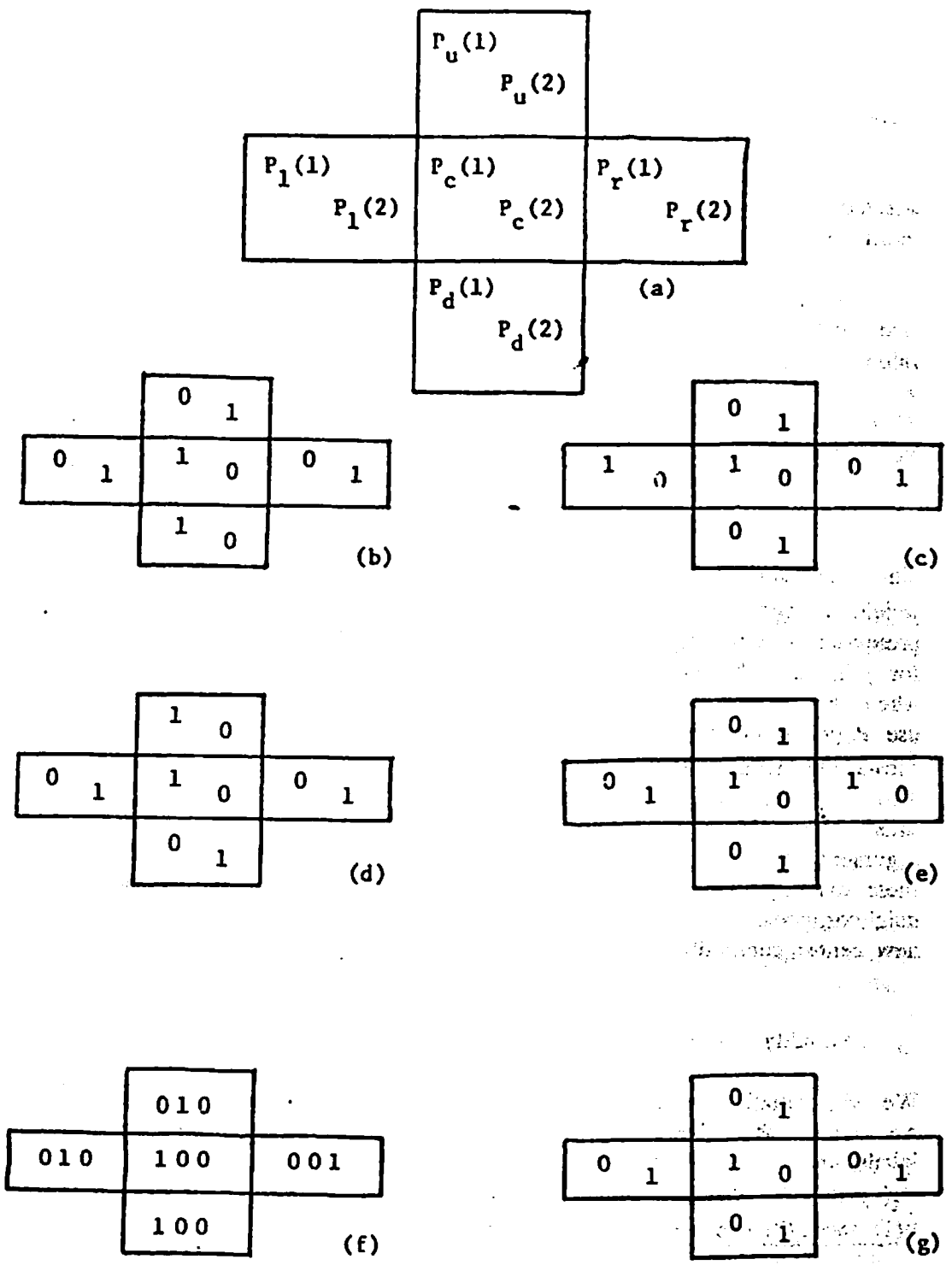


Figure 14: Center Compatibility Coefficients: Converged Example Cases.

$$q(1) = r_d(1,1) + r_c(1,1) + r_l(1,2) + r_u(1,2) + r_r(1,2) \quad 4.6$$

$$q(2) = r_d(2,1) + r_c(2,1) + r_l(2,2) + r_u(2,2) + r_r(2,2) \quad 4.7$$

As discussed above, all compatibilities for the neighbors are constants computed from the current image, thus allowing us to solve for the center compatibilities in the inequality $q(1) >= q(2)$:

$$r_c(1,1) - r_c(2,1) >= \begin{pmatrix} r_l(2,2) - r_l(1,2) + r_r(2,2) - r_r(1,2) \\ + r_d(2,1) - r_d(1,1) + r_u(2,2) - r_u(1,2) \end{pmatrix} \quad 4.8$$

The pixel which supports the central pixel could, of course, appear at any of the other neighbors and all four possible orientations must be considered. Thus, a similar derivation using the cases of figures 14c-14e yields three additional inequalities for $r_c(1,1) - r_c(2,1)$ which must be satisfied if the stability of the label geometry in any orientation is to be guaranteed. The value for $r_c(i,i)$ is arbitrarily assigned to be zero. We can then solve for $r_c(2,1)$ which simultaneously satisfies the four directional inequalities. By reversing the labels in figure 14b, we can solve for the $r_c(1,2)$ coefficient.

The Stability Constraint - The Multi-label Case.

Extension to multiple labels is fairly straightforward. Note that in inequality 4.8 each neighbor introduced exactly two terms for the $q(1) >= q(2)$ inequality. Figure 14g is an example where a third label is seen in the neighborhood. The same constraint inequality ($q(1)>=q(2)$) can be used to derive additional constraints on $r_c(1,2)$:

$$r_c(1,1) - r_c(2,1) >= \begin{pmatrix} r_l(2,3) - r_l(1,3) + r_r(2,2) - r_r(1,2) \\ + r_d(2,1) - r_d(1,1) + r_u(2,2) - r_u(1,2) \end{pmatrix} \quad 4.10$$

As can be seen, the only terms which have changed (between inequalities 4.8 and 4.10) correspond to the neighbor with the third label. It is clear that the maximum of these term pairs would be the more restrictive constraint. Let us define L as the set of all n labels and LX as the set of all n labels excluding the best label of the central pixel, then the maximum constraint imposed on the $q_1 >= q_2$ inequality by neighbor x would be:

$$\underset{z \in LX}{\text{MAX}} \; (r_x(2,z) - r_x(1,z)) \qquad 4.11$$

Generalizing to other neighbors with different labels produces the following constraint:

$$r_c(1,1) - r_c(2,1) >= \left(\begin{array}{l} \underset{z \in LX}{\text{MAX}} \; (r_l(2,z) - r_l(1,z)) \\ + \underset{z \in LX}{\text{MAX}} \; (r_r(2,z) - r_r(1,z)) \\ + r_d(2,1) - r_d(1,1) \\ + \underset{z \in LX}{\text{MAX}} \; (r_u(2,z) - r_u(1,z)) \end{array} \right) \qquad 4.12$$

As above, there are still a total of four constraints for each label pair (one for each possible direction of support, corresponding to figures 14b - 14e).

Given our approach to the specification of compatibility coefficients, all the coefficients on the right hand side of equation 4.12 are image-dependent and calculated directly from the joint probabilities of spatially adjacent labels. Then, combining all image dependent constants in 4.12 into the image dependent constant C_2 and assuming $r_c(i,i) = 0$, one can rewrite inequality 4.12 as:

$$r_c(2,1) <= C_2. \qquad 4.13$$

Effectiveness Constraint.

We must also ensure that the effectiveness constraint, which guarantees that isolated unsupported labels should not remain at convergence, is satisfied as well. In this case we want the updated value of P_c to decrease. From equation 3.3 we can see that this is true whenever

$$\frac{1 + q(i)}{\underset{j \in L}{\Sigma} P_c(i) * (1 + q(j))} < 1 \qquad 4.14$$

Again, the denominator of this form is the weighted average of $(1 + q(j))$ for all j in L. If we have $q(i) < q(j)$ for all j then this form will be less than one and the central pixel will not remain converged at label i. To satisfy this constraint in the two label cases such as that shown in figure 14g we must have $q(1) < q(2)$. In this case, we have

$$q(1) = r_c(1,1) + r_d(1,2) + r_l(1,2) + r_u(1,2) + r_r(1,2) \quad 4.15$$

$$q(2) = r_c(2,1) + r_d(2,2) + r_l(2,2) + r_u(2,2) + r_r(2,2) \quad 4.16$$

Again, solving for the center compatibilities we obtain:

$$r_c(1,1) - r_c(2,1) < \left(\begin{array}{c} r_l(2,2) - r_l(1,2) + r_r(2,2) - r_r(1,2) \\ + r_d(2,2) - r_d(1,2) + r_u(2,2) - r_u(1,2) \end{array} \right) \quad 4.17$$

This form may also be generalized to the multiple label case (in the same manner as the previous case) yielding the inequality:

$$r_c(1,1) - r_c(2,1) < \left(\begin{array}{l} \underset{z \in LX}{\text{MIN}} \; (r_l(2,z) - r_l(1,z)) \\ + \underset{z \in LX}{\text{MIN}} \; (r_r(2,z) - r_r(1,z)) \\ + \underset{z \in LX}{\text{MIN}} \; (r_d(2,z) - r_d(1,z)) \\ + \underset{z \in LX}{\text{MIN}} \; (r_u(2,z) - r_u(1,z)) \end{array} \right) \quad 4.18$$

Again, under the assumption that the $r_x(i,j)$ compatibilities are image dependent constants, the right side of this equation (including the $r_c(1,1)$ term) may be combined into the image dependent constant C_1 and rewritten as:

$$r_c(2,1) > C_1 \quad 4.19$$

This, together with the result of the previous section, yields a bounded interval in which $r_c(2,1)$ must lie in order to satisfy both constraints:

$$C_1 < r_c(2,1) <= C_2. \quad 4.20$$

For $r_c(2,1)$ less than C_1, an isolated label could remain converged through the relaxation process, while for $r_c(2,1)$ greater than C_2 one pixel wide structures could be destroyed. When $r_c(2,1)$ is close to C_1, the relaxation has strong inertia (the

magnitude of the center pixel's influence is large) and the relaxation will barely keep isolated pixels from converging. When $r_c(2,1)$ is close to C_2 the relaxation has less inertia and the relaxation will barely maintain converged, one pixel wide, fine structures. At the weak extreme the relaxation may not change the labeling of some incorrectly labeled pixels while at the strong extreme fine structure might be obliterated long before local convergence is reached. Since the actual values of C1 and C2 are dependent on the image, the interval may in fact be empty for some images and cluster set combinations, making it impossible to simultaneously satisfy both constraints.

Partially Converged Neighborhoods.

One can gain further understanding of the behavior of the updating process for different values of $r_c(2,1)$ in the interval $(C_1 , C_2]$ by considering some slightly more complex cases in which local convergence is not complete. In figure 15a the three neighbors supporting label 2 are converged, yet the central pixel and the pixel supporting the current label of of the central pixel have a probability x for label 1 and a probability of $\bar{x} = 1 - x$ for label 2:

$$P_c(1) = P_d(1) = x, \; x > .5$$

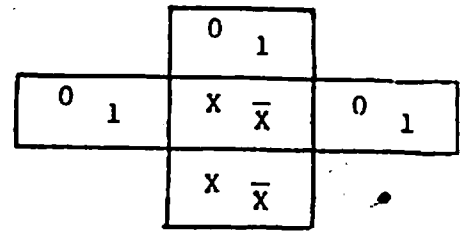
$$P_c(2) = P_d = \bar{x}$$

In order to satisfy the stability constraint we again want $q(1) >= q(2)$. Here, q(1) and q(2) are somewhat more complex:

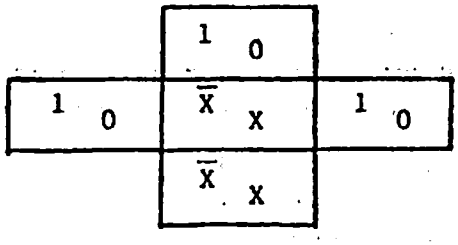
$$q(1) = x\, r_d(1,1) + \bar{x}\, r_d(2,1) + x\, r_c(1,1) + \bar{x}\, r_c(2,1) + r_l(1,2) + r_u(1,2) + r_r(1,2) \qquad 4.21$$

$$q(2) = \bar{x}\, r_c(2,2) + x\, r_c(1,2) + \bar{x}\, r_d(2,2) + x\, r_c(1,2) + r_l(2,2) + r_u(2,2) + r_r(2,2) \qquad 4.22$$

Substituting into the inequality $q(1) >= q(2)$ and rearranging terms we can obtain



(a)



(b)

Figure 15: Partial Convergence Case for Center Compatibility Coefficients.

$$\begin{aligned} & x\,(r_c(1,1) - r_c(1,2)) \\ & -\bar{x}\,(r_c(2,2) - r_c(2,1)) \end{aligned} \quad >= \quad \begin{aligned} & \bar{x}\,(r_d(2,2) - r_d(2,1)) \\ & + x\,(r_d(1,2) - r_d(1,1)) \\ & + r_l(2,2)-r_l(1,2) \\ & + r_u(2,2)-r_u(1,2) \\ & + r_r(2,2)-r_r(1,2) \end{aligned} \qquad 4.23$$

Again, there are a total of four directional cases to consider and the inequality can be extended to the multiple label case. Collecting the constant terms of the resulting form into a single constant C_3, we can write this form as

$$\bar{x}\, r_c(2,1) - x\, r_c(1,2) >= C_3. \qquad 4.24$$

Note that now, the computations of the $r_c(2,1)$ coefficient and the $r_c(1,2)$ coefficient are no longer independent. By considering the complementary case, shown in figure 15b (the only other case which also affects this pair of compatibilities), we obtain:

$$\bar{x}\, r_c(1,2) - x\, r_c(2,1) >= C_4. \qquad 4.25$$

Solving these two simultaneous inequalities is straightforward.

This methodology provides a powerful tool both for controlling the behavior of the relaxation algorithm and for understanding its behavior and limitations.[13] Given the previous derivations one can compute the following values:

(1) Given the compatibility coefficients which just satisfy the effectiveness constraint one can compute the lowest probability x which can be guaranteed to preserve one pixel wide regions.

(2) Given that one pixel wide regions of label probability x are to be preserved, one can find the compatibility coefficients which satisfy the stability constraint for this case. It may not be possible to satisfy the effectiveness constraint for low values of x.

One can choose to abandon the effectiveness constraint in order to guarantee that certain fairly weak one pixel wide regions will be preserved. In this case, one can determine under what conditions an isolated pixel might remain converged (cases in which the effectiveness criterion fails).

---

[13] An important beneficial side effect of this understanding is that it allows for a much more efficient implementation on a sequential machine. No relaxation needs to be performed at pixels which are converged to label i and which have at least one neighbor which is also converged to label i. After the first few iterations of relaxation the vast majority of the pixels in the image fall into this category resulting in a significant decrease in the computational cost per iteration of relaxation.

Empirical Findings.

The following two-label experiments (presented in figure 16) compare the proposed center compatibility formulation with that used in Nagin. The experiments show that the behavior of the Nagin formulation, in a converged neighborhood, is a function of the initial probability assignments while the proposed formulation results in the same behavior regardless of the initial probability distribution.

Figures 16a - 16d show the initial probability distribution for two examples. The two examples both consist of a 64x64 background labeled 2 with a 32x32 foreground square labeled 1 and some fine image structure labeled 1 in the upper left corner. It is the behavior of this fine structure during the iterative updating that is of interest. The only difference between the two examples is that in the first example initial probabilities are close to convergence to begin with (initial probabilities of .95 and .05 for the two labels) while in the second example initial probabilities are almost the same (.48 and .52 for the two labels). Figure 16a shows the probability of label 1 for the first example while figure 16c shows the probability of label 1 for the second example. Figures 16b and 16d show the corresponding probabilities of label 2. Since P(2), the probability of label 2, is always 1 - P(1), we will show only the probability of label 1 in the remainder of this presentation.

The initial probabilities of figures 16a and 16b are used to compute a set of compatibility coefficients for the first example while figures 16c and 16d are used for the second example. The relaxation update is then applied to a small image corresponding to the fine structure in the two examples above. In this experiment all of the labels in figure 16e are converged. The square in the lower right portion of figure 16e corresponds to a single pixel and should be deleted since it is an isolated label with no support in its neighborhood. The rectangle to the left of the square consists of two pixels and should be maintained by the relaxation according to our goals. The vertical one pixel wide structure above the square should also be maintained and certainly the large multi-pixel structure in the upper left should be maintained.

The results of applying the Nagin algorithm to the test image of figure 16e is shown in figures 16f and 16g. Figure 16f shows the result after 100 iterations of relaxation using the compatibility coefficients computed from figures 16a and 16b. Figure 16g shows the corresponding result for the compatibilities computed from figures 16c and 16d. As can be seen in figure 16f, when the initial probabilities of labels were large the relaxation destroyed the one pixel wide image structures. On the other hand, figure 16g shows that when the initial probabilities were especially ambiguous, the Nagin relaxation preserves even one pixel regions.

Figures 16h and 16i show the corresponding results using the modified center compatibilities. As can be seen, in both cases, the resulting labelings are identical to each other and satisfy both of the constraints defined above. The one pixel region

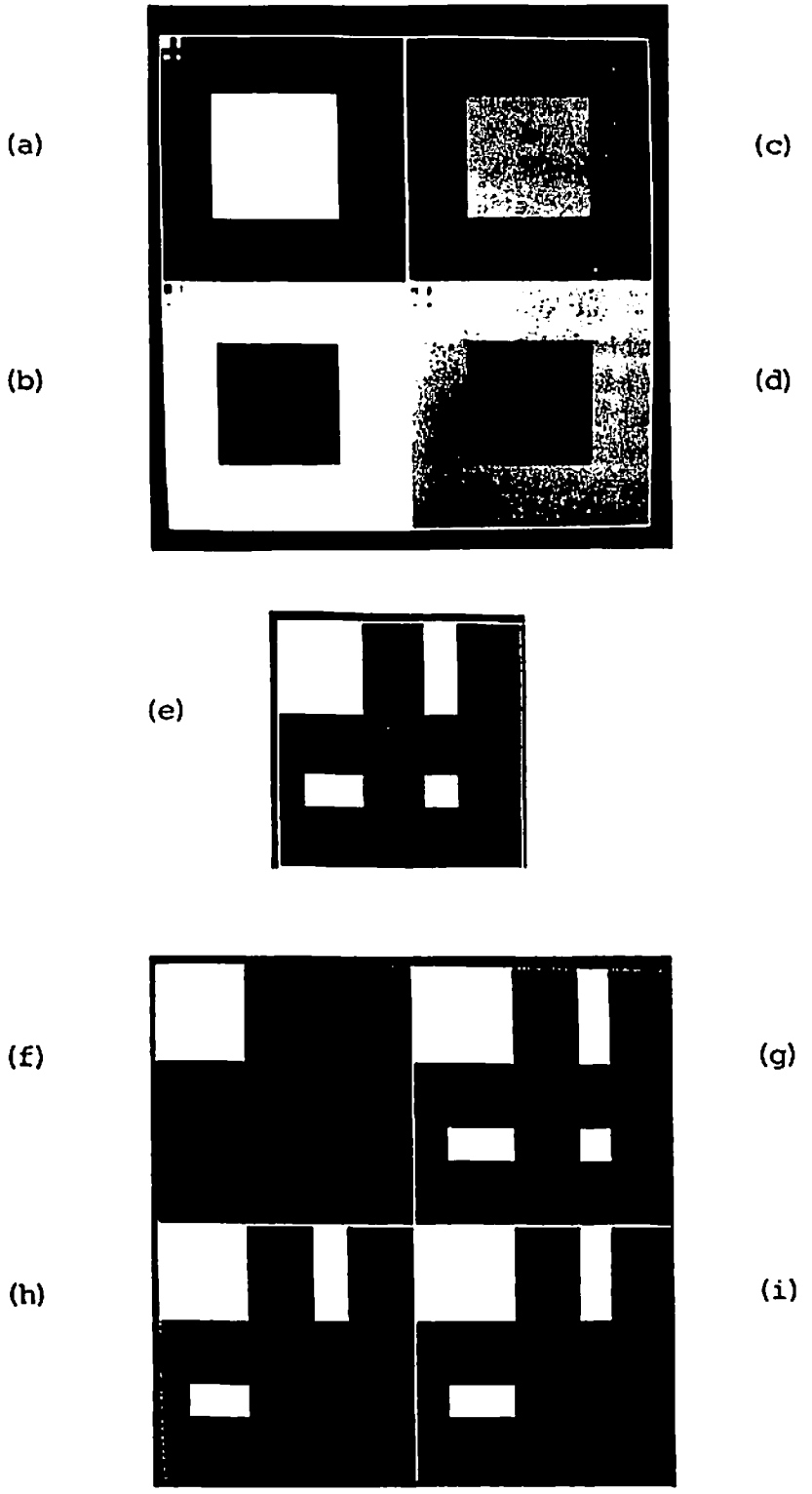


Figure 16: Center Compatibility Modification - Experimental Results.

was correctly deleted and the one pixel wide structures were preserved regardless of the initial probability distribution of the labels.

5.0 EVALUATING SEGMENTATION ALGORITHMS

5.1 Issues of Evaluation -- Introduction

For all but the simplest scenes, a large number of alternative segmentations are possible. Which of these alternatives is "the best segmentation" is a difficult, and in most cases, an unanswerable question. However, in the context of a system which utilizes multiple segmentation processes, that produce many different segmentations, some attempt must be made to deal with this difficult issue. Figure 17a, b, and c show three different segmentations of the scene in figure 17a. Which of the segmentations is best and why? Clearly the goal of the processing is a critical factor here.

This short chapter does not attempt to solve the difficult problems of segmentation evaluation nor does it make any theoretical claims. Rather, it delineates the issues involved in evaluating segmentation algorithms, considers alternative methodologies for evaluation, and explicates the methodology used in this thesis.

The issue of segmentation has not been ignored by the image processing community; a session of the 1979 Pattern Recognition Conference entitled "Scene Segmentation and Interpretation" contains a number of papers which present segmentation results for a common image. This is a step in the right direction since at least the image processing algorithms could then be compared on the same image. However, the evaluation of the segmentations was entirely subjective.

Papers which address the problems of segmentation evaluation include Nagin, Kohler, Hanson, and Riseman [NAG79], [NAG80], and [NAZ83].

5.2 Issues of Evaluation -- Pattern Classification Paradigm

One standard methodology used in the field of pattern classification involves training the classifier on a set of training patterns, and then evaluating the classifier on a different set of test patterns. The training process is generally an optimization procedure which attempts to minimize the probability of incorrect classifications. In the supervised approach, the correct classification of all training patterns is known. Typically, either the form of the family of distributions is assumed or the form of the discriminant function is assumed, and only the parameters of the distributions or discriminant function need to be optimized. This methodology permits the objective comparison of classifiers using error rate estimates computed from experiments using the patterns in the test set. It is important that the test set of patterns typify, and in some sense model, the patterns which will actually be encountered in practice. It

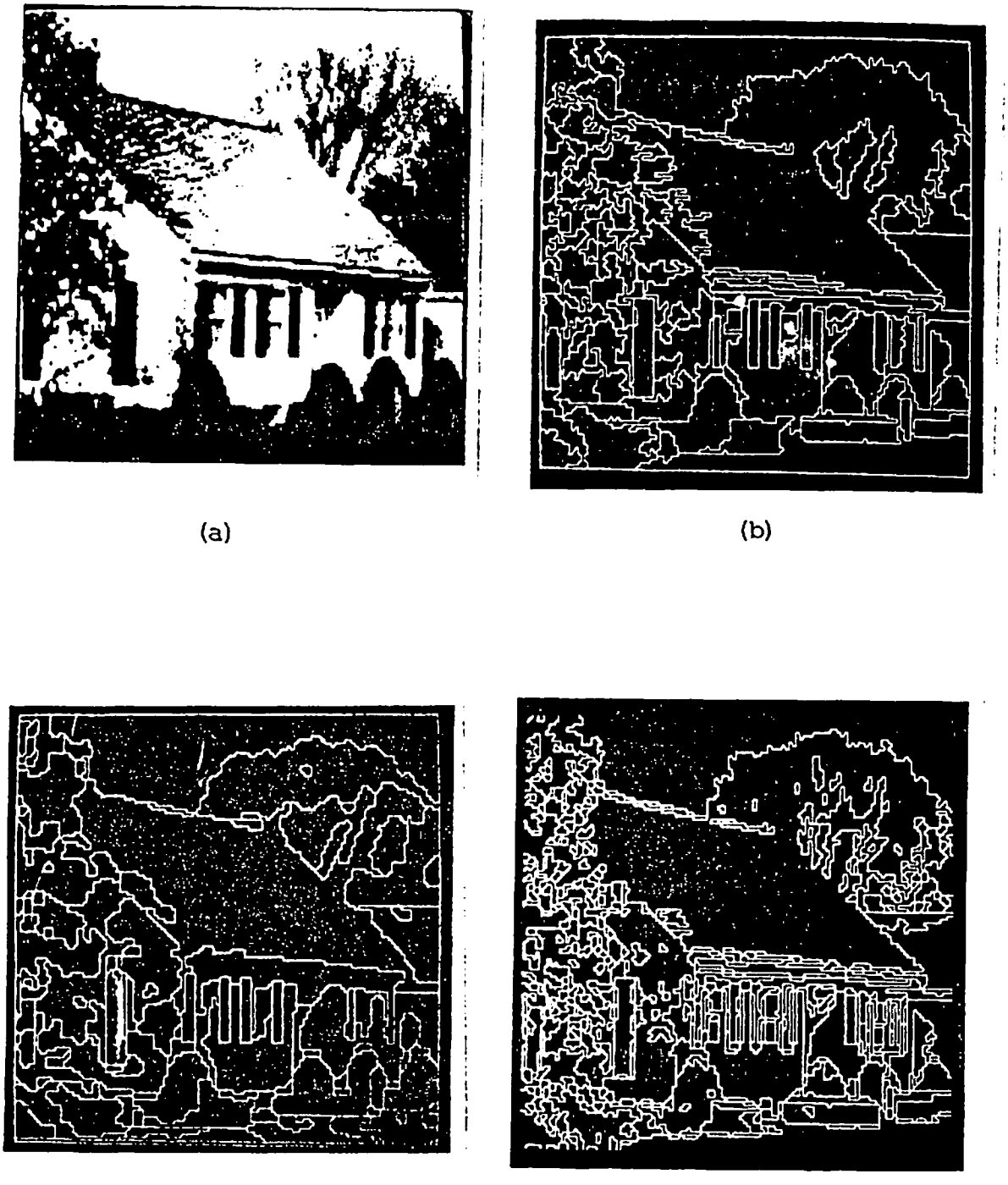


Figure 17: Alternative Segmentations of a House Scene.

is also important that the patterns represent an abstraction of the actual patterns so that the training phase will not bias the classifier to anomalies present only in the training set. The question asked in the evaluation phase is how well will this classifier perform in general? Unfortunately, due to a variety of difficulties, this methodology has not generally been applied in the field of image segmentation.

A number of practical factors, including the availability of digitized images and the tremendous per image processing cost of segmenting even simple scenes make the statistical optimization approach difficult in the image analysis domain. Many published image segmentation algorithms have only been applied to a very small number of images. Furthermore, often those same images were used during the algorithm development phase. This means that the algorithm is evaluated on a test set that is identical to the training set. This makes any generalization as to the power of the published algorithm for a certain class of images questionable since the algorithm may be sensitive to anomalies of the small set of training images.

Applying the pattern classification paradigm to image analysis suffers from more fundamental problems. In scene segmentation the correct segmentation is not generally known. An important consideration in evaluation of a segmentation is the global purpose of the segmentation. In cases where the goal is well defined, such as counting blood cells in a given field, the segmentation algorithm may be indirectly evaluated according to the global result: did the algorithm detect the proper number of blood cells, how many cells were missed, and how many false alarms occurred. Errors may not be uniquely attributed to the segmentation process since the errors might be due to the classification component or some interaction between the segmentation and classification components. When the goals of a segmentation algorithm are well defined and the results can be quantitatively compared to accepted solutions, then the pattern recognition paradigm is applicable.

When segmentation algorithms are applied to more complex images such as natural outdoor scenes and the goal is to build an accurate three dimensional model of the scene, this evaluation methodology becomes much less viable. Building a three dimensional model from the segmentation is a complex and poorly understood task. Comparison of any models produced using different segmentation algorithms would require some evaluation function for comparing differences between the real world and the constructed models. Such an evaluation function would have to determine which objects in the scene were correctly detected and how accurately the objects were located, as well as which objects in the scene were missed and which objects were incorrectly included. Clearly, such an evaluation function would be quite complex and dependent on the goals of the external system which requests the segmentation. Again, comparison of alternative segmentation algorithms is complicated by the potential interaction between the segmentation process and the semantic interpretation process. Tuning the interpretation component to each segmentation process might be required. A major practical constraint preventing experimentation with the use of such a "high level" evaluation paradigm is the absence (at this time) of an automatic interpretation system that can be easily run on many images.

In conclusion, it is desirable to evaluate segmentations in the global context of the processing goal directly, but this evaluation in context is not currently feasible except for very simple and well understood images. This implies that some attempt must be made to evaluate segmentation algorithms directly based on the segmentations produced.

5.3 Possible Evaluation Methodologies

Segmentations have generally been evaluated subjectively on one or two images. Although this may lead to insights into the power or limitations of a particular algorithm, this evaluation methodology is not adequate for the quantitative comparison between two similar segmentation algorithms applied to a large set of test images. As argued above a quantitative evaluation based on the global result of the segmentation process is desirable but not very feasible at this time. The remaining alternative is to evaluate the two dimensional segmentation directly. In order to quantify the evaluation of a segmentation, one might compare the segmentation against a registered model of the scene. The pixel misclassification rate, or frequency of edge errors, could be used to evaluate the segmentation. The use of such simple misclassification rates does not always capture all of the error information desired. Sometimes a single misclassified pixel may alter the connectivity structure of the image, while other misclassified pixels might not; an additional boundary segment may be superfluous or may separate a single correct region into two parts. Thus, a more complex evaluation function based on the global utilization of the segmentation will eventually be desirable. The more immediate problem is: where is the correct reference model to come from? Human generated hand segmentations are one possible source. Using segmentations produced by humans as a model may introduce unrealistic expectations since these may embed implicit biases and expectations, knowledge about object shapes, sizes, and colors, and knowledge about perspective, shading, texture, and depth.

One way to attack the difficult problem of obtaining goal segmentations is to consider several different classes of images along a complexity dimension. At the most complex extreme one finds the natural outdoor scenes with all of the difficulties implied by natural light, complex shapes, and textured surfaces; while at the other extreme one finds very simple scenes with no noise, no texture, and none of the complex intensity gradients that make segmentation difficult. It may be impossible to define a correct segmentation in the outdoor scenes, but a correct segmentation may be easily and unambiguously defined for the very simple scenes.

Consider a very simple black and white image of a bright square superimposed on a dark uniform background. In this simple image there can be no ambiguity about the correct segmentation. All segmentation algorithms should produce a perfect segmentation of this simple scene. Unfortunately, the results on such a simple scene cannot be generalized to the domain of complex outdoor scenes.

5.4 Evaluation Methodology

One can generate more complex scenes along a continuum by adding those image characteristics found in natural scenes which make them difficult to segment. The presence of (let us say Gaussian) noise due to the photographic and digitization processes is one of these characteristics. Some of this noise is due to the transformation of the continuous world into the spatially discrete representation with discrete feature values. The importance of problems due to certain forms of noise are sometimes overstated, since these problems may be reduced by increasing the hardware quality (better lenses, larger format negatives, or higher resolution scanners) or by simple image enhancement operators [ROS76b], [PRA79], [PRA78]. The other characteristics of the natural outdoor scene which pose problems are due to photometric and geometric considerations. These characteristics include intensity gradients, shadows, highlights, and surface texture.

By modeling these characteristics one can introduce them individually or in groups into the simple images to obtain more complex images based on the same physical model. This continuum of artificial images provides a powerful tool both for segmentation algorithm development and evaluation. In the development of segmentation algorithms one can begin with simple images and add one element of image complexity at a time. Behavior of the algorithm when applied to each of these characteristics helps to pinpoint the strengths and weaknesses of the algorithm, and failures of the algorithm often suggest modifications which might overcome some of the weaknesses.

With the relatively simple images competing alternative segmentation processes can be evaluated quantitatively using simple pixel misclassification methods. As the images become more complex and approach natural scenes in complexity, the same problems of evaluation as discussed for natural scenes occur and quantitative evaluation becomes less meaningful.

In this dissertation algorithms are **quantitatively evaluated** and developed using **simple constructed images**. The test images are made more complex by introducing various image characteristics that are found in natural images. The final algorithms are then demonstrated on **complex outdoor scenes** and the segmentations are **qualitatively evaluated.**

6.0 NON-SEMANTIC KNOWLEDGE IN THE SEGMENTATION PROCESS

A complete segmentation system might contain several segmentation algorithms which could be differentially applicable to different classes of images or even to different portions of a single image. Issues of interaction between the separate algorithms is explored in chapter 7. This chapter addresses the problem of instantiating a particular segmentation algorithm. Section 6.1 defines segmentation

processes and discusses their generation in general terms. The remainder of the chapter applies these general techniques to the modified Nagin segmentation algorithm. By examining particular types of image segmentation problems, we demonstrate the value of combining multiple types of non-semantic knowledge to improve the segmentation process. In particular, we address issues of undersegmentation due to missed clusters and oversegmentation due to clusters derived from micro-texture, intensity gradients, and "mixed pixel" regions in the modified histogram cluster based segmentation algorithm. Addressing these issues requires us to deal with the relationship between feature space analysis and local and global characteristics of image space.

6.1 A Segmentation Process

A segmentation algorithm consists of a set of image operators embedded in a control structure. Let us assume that it has been decided that a particular segmentation algorithm is appropriate for a given image. In order to apply the segmentation algorithm to the image, a number of important decisions must be made. The set of decisions may be split into two parts: the selection of the image features which serve as input to the algorithm, and the selection of appropriate values for any of the parameters needed by the segmentation algorithm. A segmentation process is then defined to be the instantiation of a segmentation algorithm by selection of the appropriate parameters and image features such that the algorithm may be applied to a particular image.

The selection of parameters is often assumed to be integral to the algorithm itself. Consider an algorithm which segments images by thresholding, where selection of the threshold is based on the global histograms of the intensities in the images. Is the selection of the threshold part of the segmentation algorithm or should it be considered independently? Is the use of the intensity feature an integral part of the algorithm or could a more intelligent feature selection decision (based on the particular image to be segmented) be isolated from the algorithm?

A segmentation algorithm represents a specific instantiation of a set of assumptions or heuristics about properties of the image which provide useful information with respect to segmenting the image into regions. Regions are usually assumed to correspond to objects, surfaces, or visually distinct object parts in the original three dimensional scene. For any segmentation algorithm these assumptions or heuristics are often inadequate for segmenting images which somehow violate the assumptions embedded in the algorithm. It is likely that a single segmentation algorithm will not be adequate to segment all images, and that the algorithm will make segmentation errors.

The limitations imposed on the segmentation by the segmentation algorithm's assumptions or heuristics result in an environment similar to Hearsay II with its errorful and incomplete knowledge sources. The solution to this problem in Hearsay

II was to allow knowledge sources to propose hypotheses which were then supported, contradicted, or modified by other knowledge sources based on different information. Applying this general philosophy to the design of a segmentation system, any segmentation algorithm might propose a set of parameters based on a set of assumptions or heuristics. Other sources of knowledge may then modify the initial set of proposed parameters. This chapter develops one approach to the problem of determining which segmentation process should be instantiated for a given segmentation algorithm and a given image.

The information brought to bear on the initial parameter selection may be derived from semantic expectations of the contents of the scene, from syntactic characteristics of the expected segmentation, or from global measurements of the original image. This chapter will only address the latter two alternatives.

The assumptions embedded in a particular algorithm typically fail for certain image characteristics. For instance, clustering algorithms are often sensitive to intensity gradients or micro texture. These image events or characteristics result in broad or multiple clusters in the feature histogram, which is in violation of the tacit assumption that a region will appear as a single distinct cluster easily separable from the clusters corresponding to other regions. Edge detection algorithms often have difficulty with edges whose spatial scale is different from the edge detection operator used. Image and algorithm dependent parameter values are selected or modified based on a number of general assumptions about images. These include the assumptions that adjacent and very similar pixels probably belong to the same object, that strong image gradients often indicate object discontinuities, assumptions related to the expected behavior of the segmentation in terms of region sizes and shapes, as well as expectations derived from a partial interpretation of the scene.

6.2 Cluster Selection: A Parameter Selection Example

The remainder of the chapter focusses on the selection of clusters in the Nagin segmentation algorithm detailed in chapter 3. In the Nagin algorithm, clusters are initially selected from a one dimensional histogram of a single image feature[14] The given image was partitioned into arbitrary square subimages and clusters were selected independently in each of the subimages. These clusters represent a set of initial hypotheses as to the set of optimum clusters present in each subimage. The techniques proposed below attempt to modify the initial hypotheses by adding clusters which may have been missed, deleting clusters which do not seem to correspond to image structures, and merging clusters in order to satisfy certain image constraints.

[14] Note that the cluster selection might be profitably extended to higher order feature spaces, and in fact Nagin used 2-dimensional histograms. The techniques for cluster validation presented below may be easily generalized, since they are not highly dependent on the dimensionality of the feature space in which clusters are selected.

The following section examines how clusters which may have been missed in one subimage may be recovered by using consistency cues obtained from neighboring subimages. Later sections of this chapter show how characteristics of the initial rough segmentation might guide the elimination or merging of clusters. Certain image characteristics have traditionally confounded algorithms which segment based on clusters, including:

(1) Micro-texture, which can lead to serious oversegmentation into many small regions,

(2) Wide intensity gradients which can introduce artificial boundaries within essentially uniform areas,

(3) Blurred (or mixed-pixel) boundaries which can lead to long narrow regions separating "correct" regions

(4) Highlights and specularities which can result in small regions where no surface discontinuity exists. (note that these regions may provide important cues as to the surface orientation but are not really desirable in the final segmentation).

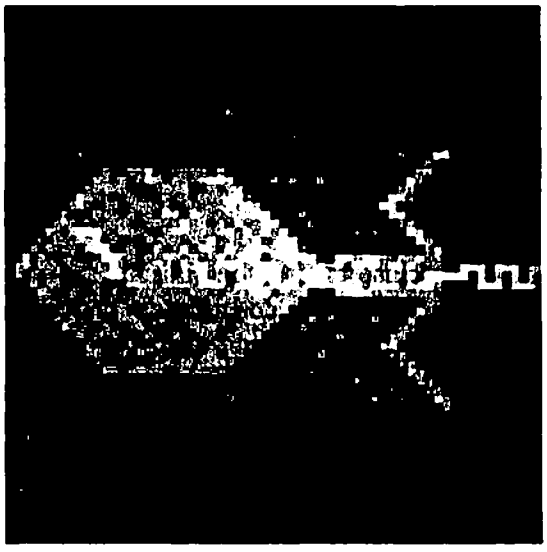
For each of these image characteristics a simple knowledge source attempts to correct the current cluster set for errors which might be due to these image characteristic.

6.3 Pre-processing

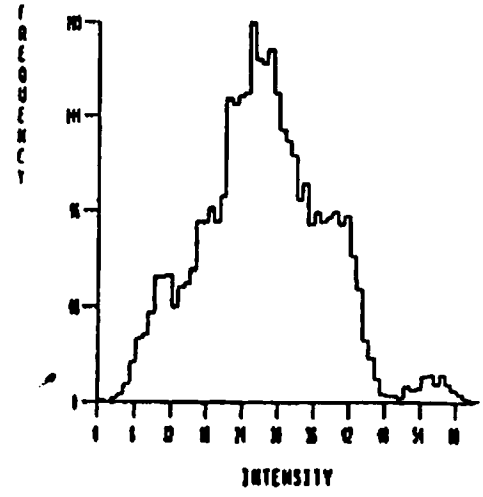
In order to increase the reliability of the initial cluster selection a noise reducing preprocessing algorithm (Overton and Weymouth 1979) was utilized. This algorithm is an iterative image smoothing function which can reduce minor image variation without destroying image structures or blurring high-contrast boundaries. The algorithm is essentially a weighted averaging process where each pixel in a three by three neighborhood contributes to the central pixel's update in direct proportion to its similarity with the central pixel and at a rate inversely proportional to a local variance of intensity about each neighbor.

When this algorithm is applied to an image, the feature histogram of the processed image often shows much better separation of clusters than the feature histogram of the unprocessed image. Figure 18 shows an example of the effect of five iterations of this preprocessing. The upper left of the figure shows the original image while the lower left shows the smoothed image. The corresponding histograms are shown to the right of the images.

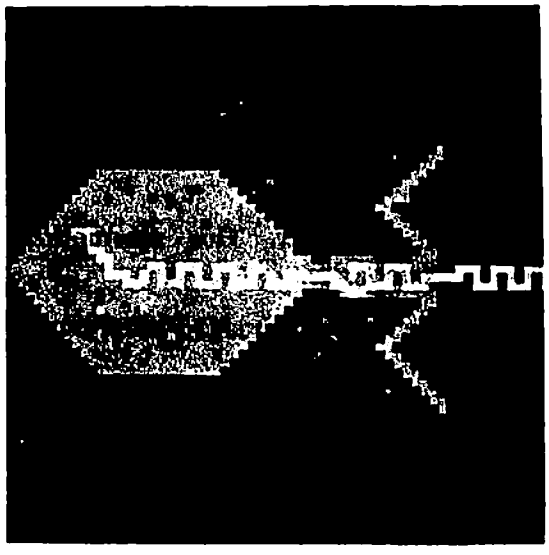
6.4 Cluster Addition Based on Subimage Consistency



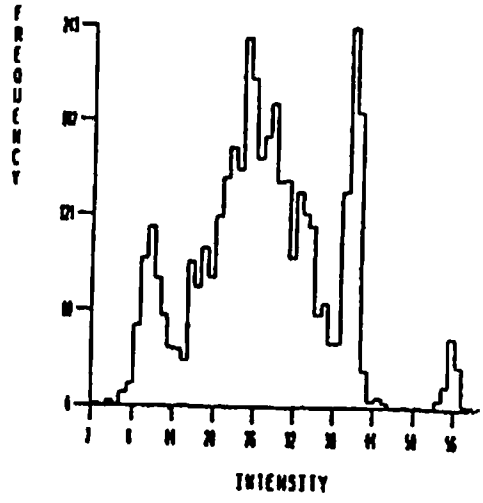
(a)



(b)



(c)



(d)

Figure 18: Overton and Weymouth Image Smoothing Example.

In order to reduce the possibility of missing clusters that correspond to relatively small regions split between two subimages, Nagin computed the feature histograms using overlapping subimages. This reduces the possibility of missing a cluster which falls half in one subimage and half in the neighboring subimage, but if the cluster is somehow obscured in either subimage, inconsistent cluster sets may result.

This section addresses the problem of missing clusters in the initial set of clusters by providing a mechanism for cluster addition based on consistency between cluster sets in adjacent subimages. Clusters can also be filtered and tuned to improve consistency with both the data in the subimage and the clusters to be added.

Inconsistent Cluster Set Example.

The mechanism for cluster selection proposed here utilizes the expected consistency between the cluster sets of adjacent subimages. Figure 19 shows a boundary between two adjacent subimages in a hypothetical example. In the left subimage, the two clusters at intensity i=10 and i=30 were found. These two clusters in the feature space give rise to the two regions shown in the left subimage. In the right subimage a single cluster at intensity i=12 was found. The right subimage is therefore uniformly labeled with the cluster at intensity i=12. Regions corresponding to each of the three clusters abut the artificial subimage boundary.

Given the spatial distribution of the regions at the subimage boundary (figure 19) a number of hypotheses about the corresponding clusters from the two subimages are possible:

(1) The clusters in both subimages are distinct and correct. This assumption implies that boundaries a and b are real image boundaries.

(2) Cluster i=10 and the cluster i=12 (in the separate subimage histograms) correspond to the same image population. This implies that:

   a. the boundary "b" is a correct boundary; or

   b. the right subimage is missing a cluster in the vicinity of i=30; or

   c. The cluster i=30 is an incorrect (superfluous) cluster which should be deleted (i.e. no region with i=30 really exists)

(3) Cluster i=30 and the cluster i=12 (in the separate subimage histograms) correspond to the same image population. This implies that:

   a. the boundary "a" is a correct boundary; or

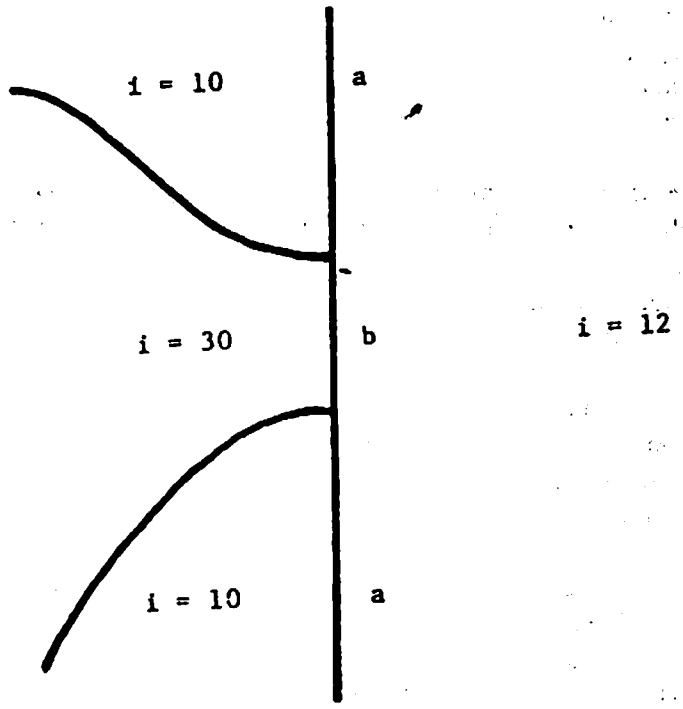


Figure 19: An Example Boundary between Subimages.

b. the right subimage is missing a cluster in the vicinity of i=10; or

c. The cluster i=10 is an incorrect (superfluous) cluster which should be deleted (no region with i=10 really exists)

The alternatives of the third case seem to be less likely since the hypothesis that cluster i=30 corresponds to the same population as the cluster i=12 seems less tenable than the hypothesis that cluster i=10 and cluster i=12 represent the same population. Note that ideally, clusters from the same population would be identical in the two subimages, but in reality, sampling errors, slow spatial gradients, and other image events will result in different observed clusters in the two subimages but the difference in observed cluster locations should be relatively small.

In the case that the first hypothesis holds or hypothesis 2a holds, no change to the set of clusters for one of the subimages is necessary. In the case that hypothesis 2b holds, the cluster set of the right subimage should have a cluster around i=30 added. In the case of hypothesis 2c the cluster at i=30 should be deleted from the cluster set of the left subimage.

We discard alternative 2c since we assume that the inclusion of the cluster at i=30 was based on a considerable body of local evidence (the cluster was found in the histogram, validated, and was not deleted by any of the cluster deletion mechanisms discussed below).

The set of possible actions is thus reduced to a) leaving the cluster sets alone or b) adding a cluster near i=30 to the right subimage. The cluster addition knowledge source will propose that a cluster near i=30 be added. Later processing may validate this cluster, in which case the cluster was probably missed by the clustering algorithm. On the other hand, if the cluster cannot be validated then the cluster will be deleted again, restoring the original cluster set. This implies that the decision between the cases for which the cluster sets are correct and for which clusters must be added can be effectively deferred by adding the clusters.

Summary of Cluster Addition.

In order to increase the inter-subimage cluster consistency, as discussed above, the algorithm summarized in figure 20 was developed. The algorithm consists of two major parts. In the first phase the algorithm attempts to determine all possible cluster additions which would make the subimage consistent with neighboring subimages. The second phase filters these suggestions to eliminate clusters which have no support in the subimage and shifts cluster means slightly to develop a set of canonical clusters which may have been suggested by different subimages.

Figure 21 shows a hypothetical subimage in which clusters were found at i=12, 19, 22, and 30. We use i=x to denote a cluster found at intensity x in the feature histogram. For each neighboring subimage the clusters which correspond to

(1) For each direction locate all clusters which abbut the current subimage

Sequentially match these clusters against the set of clusters in the subimage

An abbuting cluster is considered to be matched if

(a) the cluster is within a threshold of an internal cluster.

(b) and the cluster differs from the nearest internal unmatched cluster by less than any other unmatched abbuting cluster differs from its nearest internal cluster.

(c) and the cluster match does not cross any previous matches.

This step produces a set of matched clusters as well as a set of unmatched abutting clusters.

(2) For all sets of matched clusters and sets of unmatched clusters validate the clusters by the following procedure.

(a) Group all unmatched clusters which lie within theta of another and group interior clusters with the abbuting clusters which matched them

(b) For all cluster groups, compute the relocalized histogram for that cluster group.

(c) Select the histogram mode within theta of any of the contributing clusters as the representative for that cluster group.

(d) Filter out any cluster group whose mode has a frequency of less than some threshold.

Figure 20: Cluster Addition and Validation Algorithm.

(1) For each direction locate all clusters which abbut the current subimage

Sequentially match these clusters against the set of clusters in the subimage

An abbuting cluster is considered to be matched if

(a) the cluster is within a threshold of an internal cluster.

(b) and the cluster differs from the nearest internal unmatched cluster by less than any other unmatched abbuting cluster differs from its nearest internal cluster.

(c) and the cluster match does not cross any previous matches.

This step produces a set of matched clusters as well as a set of unmatched abutting clusters.

(2) For all sets of matched clusters and sets of unmatched clusters validate the clusters by the following procedure.

(a) Group all unmatched clusters which lie within theta of another and group interior clusters with the abbuting clusters which matched them

(b) For all cluster groups, compute the relocalized histogram for that cluster group.

(c) Select the histogram mode within theta of any of the contributing clusters as the representative for that cluster group.

(d) Filter out any cluster group whose mode has a frequency of less than some threshold.

Figure 20: Cluster Addition and Validation Algorithm.

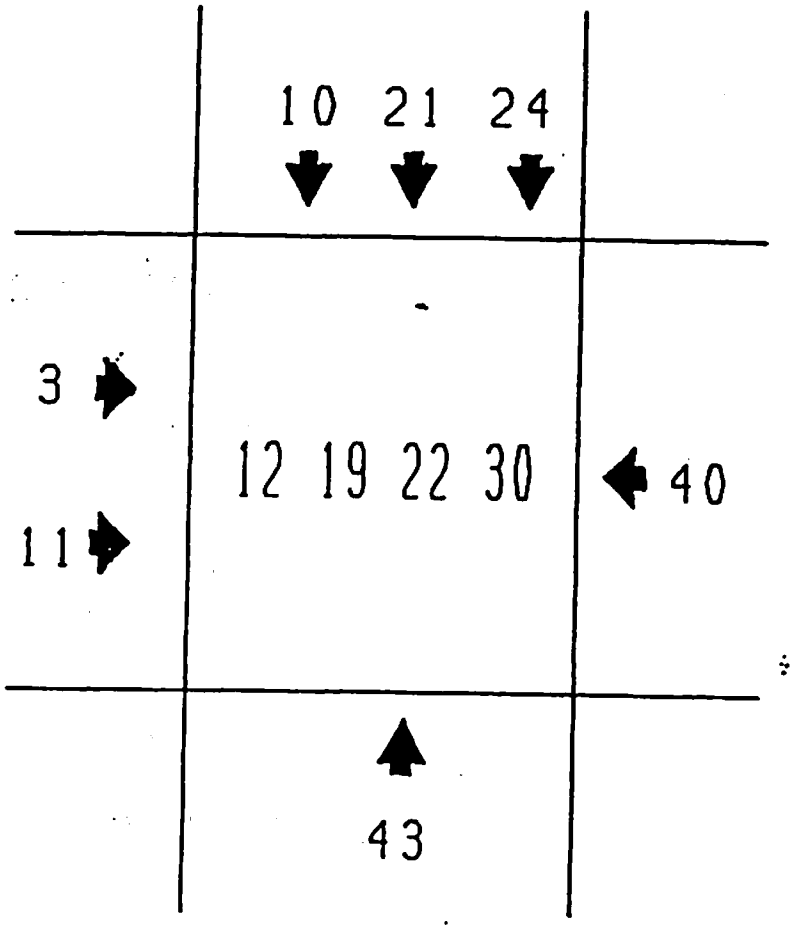


Figure 21: Subimage View of Neighboring Clusters.

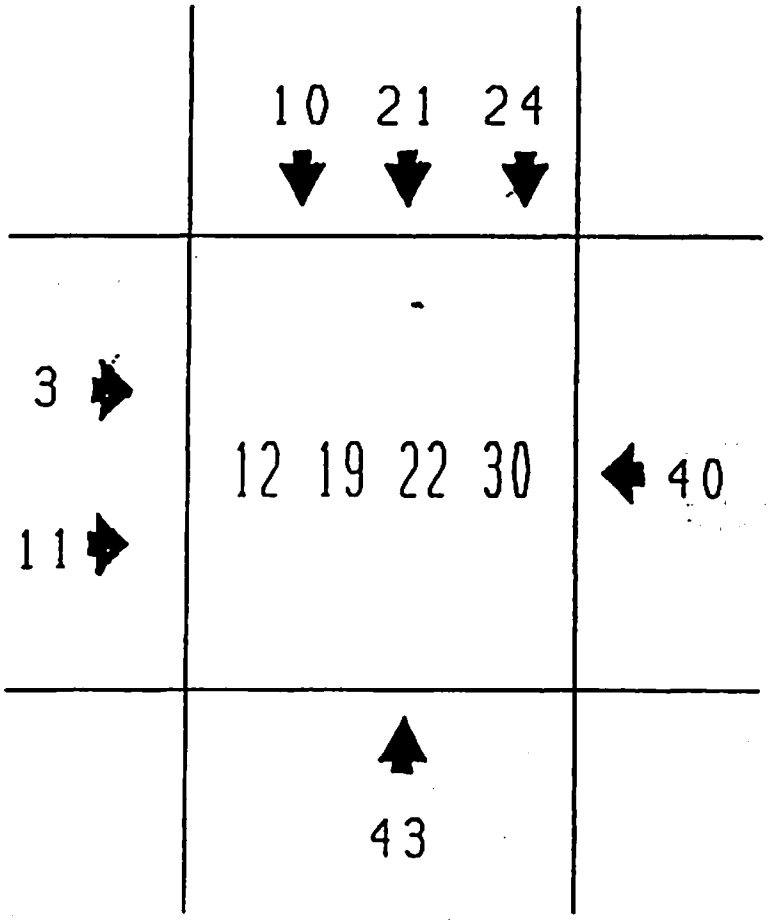


Figure 21: Subimage View of Neighboring Clusters.

the regions which abut the central subimage are matched against the current clusters.

The matching process is straightforward (see figure 22 for the case of matching the clusters abutting the subimage from above). The closest clusters which differ by less than some predefined threshold ($\theta = 5$) are considered to be matched. Matched clusters are eliminated and the next match is found until no more matches can be found. Matches are not allowed to "cross" previous matches. In figure 22, the first match pairs the labels at i=21 and i=22 while the second match pairs labels at i=10 and i=12. No further matches are possible. Matching i=24 and i=19 would "cross" the match between i=21 and i=22 since i=24 is greater than i=21 while i=19 is less than i=22. Matching i=24 and i=30 would violate the threshold restriction limiting matches to clusters whose means differ by at most five.

In the case of figure 21, the unmatched clusters would be i=24 from the subimage above, i=3 from the left subimage, i=40 from the right subimage, and i=43 from the subimage below. Furthermore, the clusters at i=10, i=11, and i=12 are matched, the clusters of i=21 above and i=22 at the center are matched, and the clusters at i=40 and i=43 are matched. The unmatched clusters which are to be added to the center subimage's cluster set would then be near i=3, i=24 and somewhere near i=40 and i=43.

The validation of clusters is based on a subhistogram of the current subimage. The subhistogram is defined for each matched cluster set as the cumulative histogram for all subimage quadrants which directly abut the boundary from which the cluster was propagated. This is simpler than it sounds, as exemplified by figure 23. The cluster i=3 was suggested by the left subimage, therefore the validation subhistogram is over quadrants 1 and 3 or the left half of the subimage. For the clusters near i=40 and i=43 the quadrants 2, 3, and 4 would define the subhistogram. This mechanism focusses on the area of the image which presumably contains support for the label to be added. The subhistograms for each cluster set are searched for clusters near the clusters proposed by the neighboring subimages (at i=3, and at i=40 and i=43 in this case). The cluster search is identical to the algorithm used to select the clusters in the first place. The clusters that are validated in this way are then added to the cluster set of the current subimage. Typically cluster sets such as i=40 and i=43 will find the same unique cluster in the subhistogram and coalesce the pair of suggested clusters into a single new cluster. Whether or not the clusters are validated, coalesced, or rejected is a function of the subhistogram and the cluster selection criteria.

Cluster Addition - Empirical Results.

The effect of the cluster addition algorithm can be seen from the following experiments. Figure 24 shows the test image for the first set of experiments. This artificially generated image depicts a T4 bacteriophage resting on an E. Coli bacterium. All of the regions in this image, except the background, have known fixed means and variances. For the background region a slow non-zero gradient is

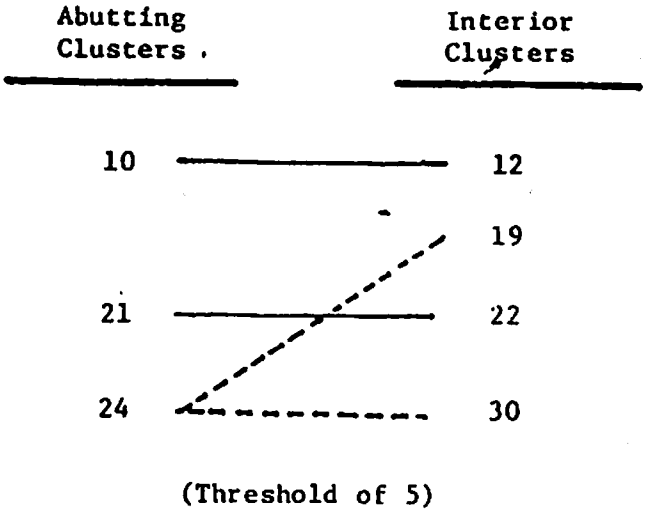


Figure 22: Cluster Matching Example.

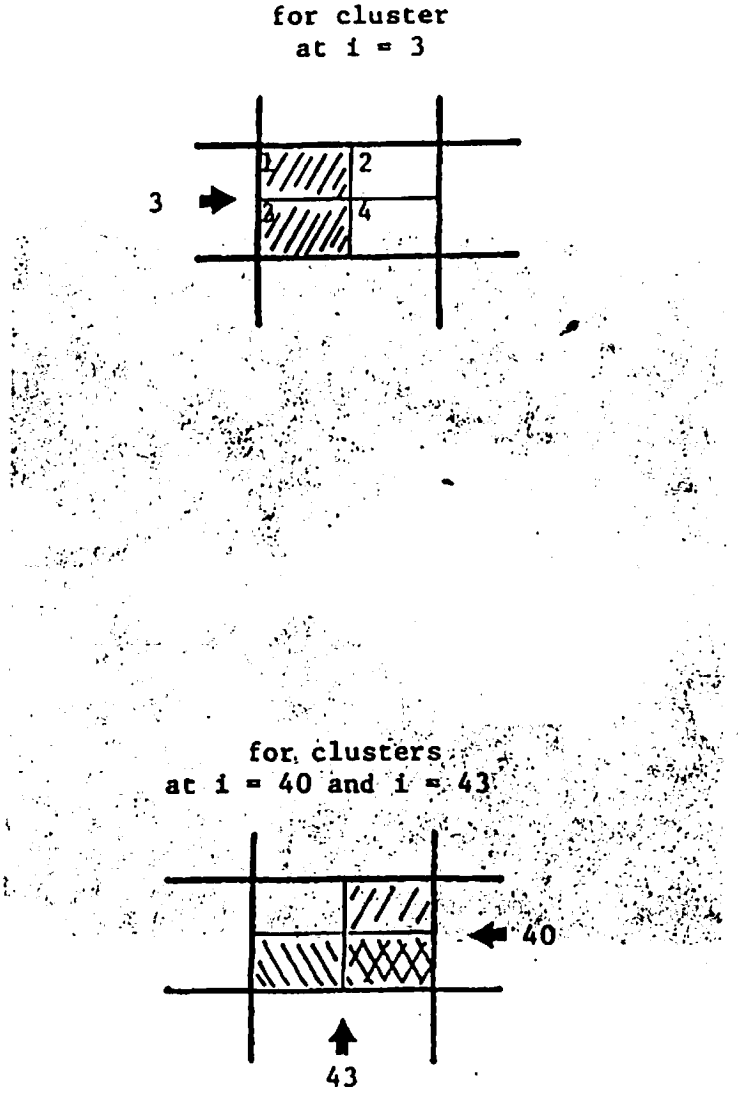


Figure 23: Subhistogram Example.

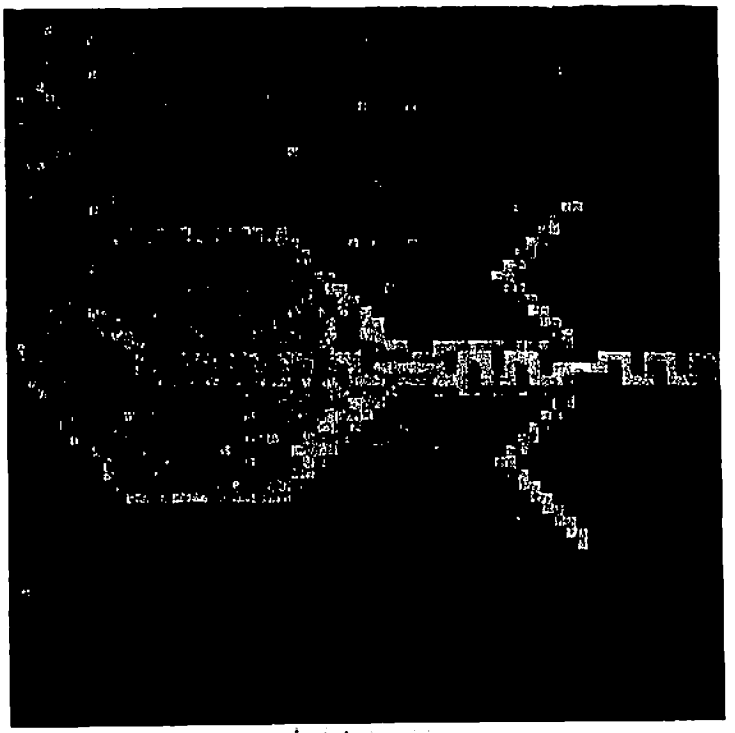
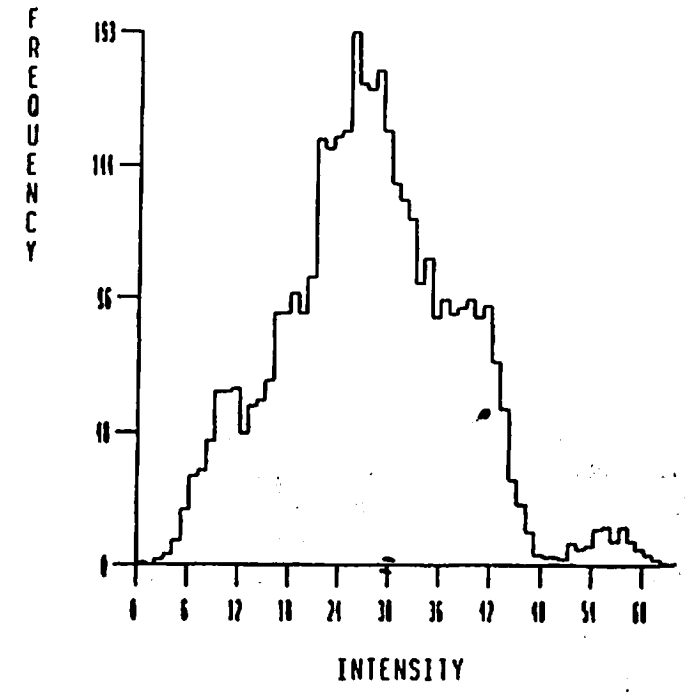


Figure 24: Virus with Gradient Test Image.

introduced such that the local mean of the background is brightest in the upper left corner and dimmest in the lower right corner. A large region with an intensity gradient is difficult for the region algorithm to capture as a single cluster since the region segmentation algorithm in effect assumes that clusters correspond to regions of uniform intensity. Note that the gradient has the effect of "spreading" out the background cluster so that other peaks are more difficult to find and the background region is at least partially labeled with clusters found within the broad background cluster (see figure 25a for the global feature histogram of this image). This results in a failure to detect some foreground regions while fragmenting the background region as seen in figure 25b.

A global segmentation of this image lead to inadequate image partitioning since the wide distribution corresponding to the large background region tended to mask the clusters corresponding to the smaller regions. A segmentation based on localization into four subimages is shown in figure 26a. As can be seen, the virus body and the background are not distinguished in the upper left quadrant and the E. Coli Cell and the background are not distinguished in the lower right quadrant. Figure 26b shows the result of applying a merging algorithm to delete the artificial boundaries introduced by the localization procedure; this merging algorithm is based upon the similarity of region features on either side of such a boundary segment and is described in section 6.6.1 below. The quality of the segmentation in figure 26 is very poor since major structures are completely lost. Figures 27a and 27b show the corresponding segmentations produced using the cluster addition algorithm to make the subimage cluster sets consistent. These segmentations are not missing the clusters which were missed in the previous figure and contain relatively fewer segmentation errors while accurately reflecting the gross structure of the original image. One problem with the segmentation is the artificial boundary due to the localization in the upper center of the image was not removed. Typically, most of the artificial boundaries left by the conservative merge algorithm can be removed using the merge algorithm described in section 6.6.2 below.

It could be argued that cluster addition was unnecessary since further localization would also locate the hidden clusters. The following example shows that this is not always the case. Figures 28a and 28b show the result of segmenting by localization to 16 subimages of 16 by 16 pixels with no cluster addition. Further localization would result in subimage histograms that are much too unreliable to permit accurate determination of clusters. In this segmentation the proportion of the image with correct clusters is increased (from 3/4 of the image with correct clusters to 14/16 of the image with correct clusters), yet two subimages with missing clusters remain. It is precisely these missing clusters that lead to the poor segmentation of figure 28b. The fact that virus body is not distinguished from the background in the first sector of the second row causes the algorithm which deletes the inter-subimage boundaries to merge the virus body and the background throughout the segmentation. Note that if a single cluster is missing in any sector then the entire object which that cluster represented may be lost when artificial boundaries are removed since the object and its surround are defined to be equivalent at that



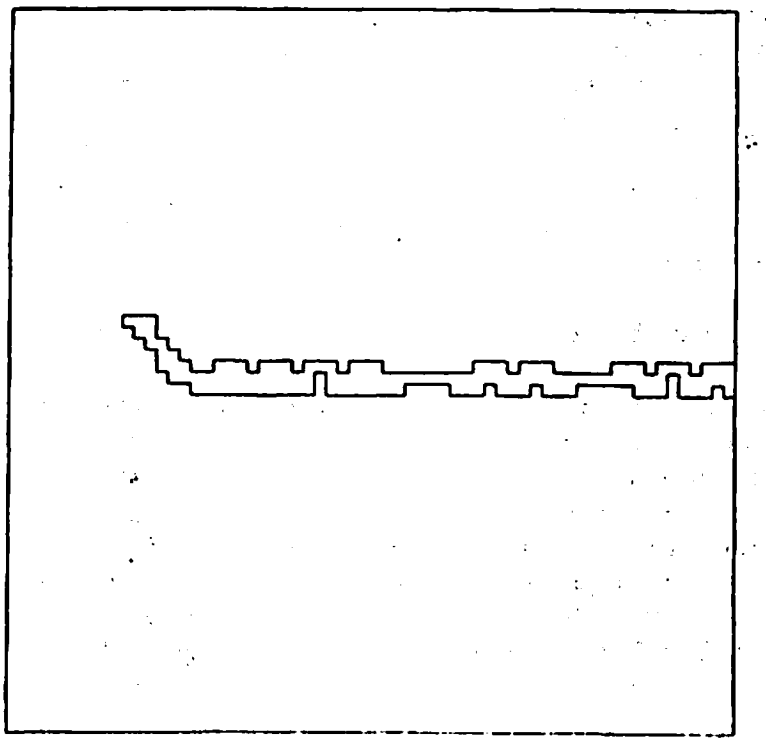
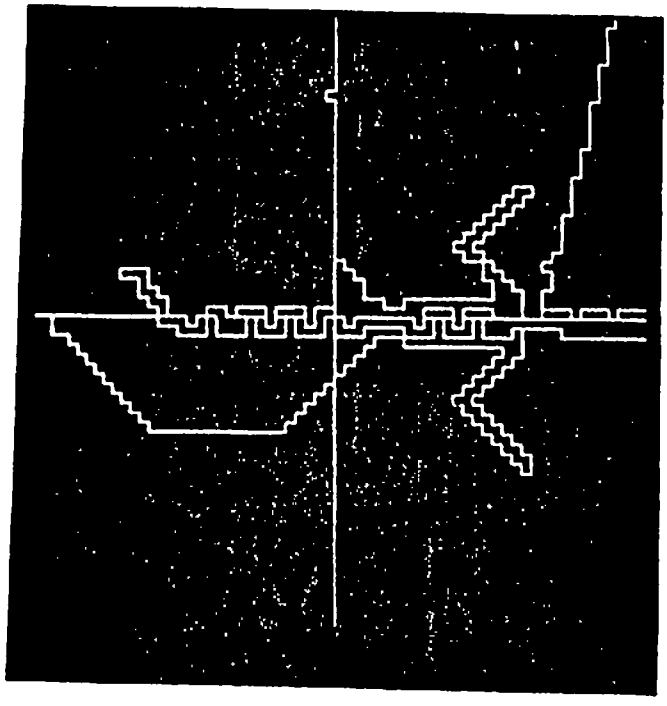
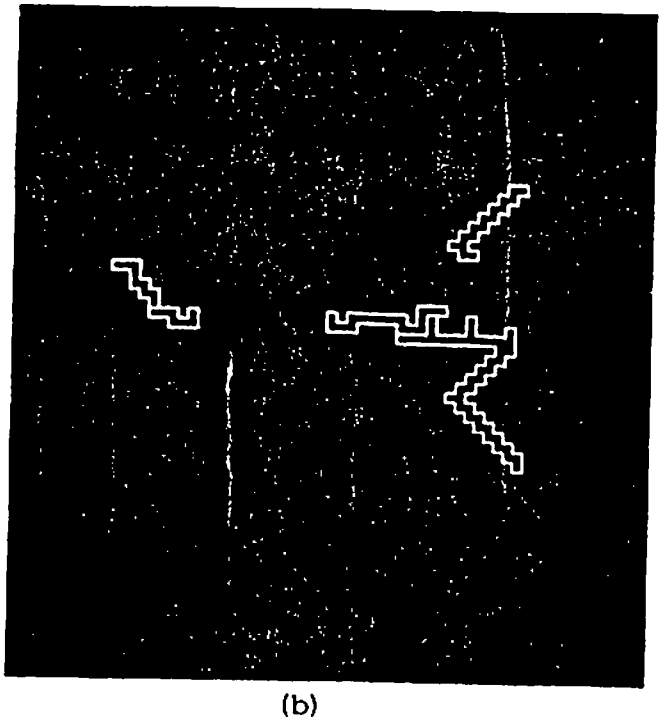


Figure 25: Global Segmentation of the Virus Image with Histogram.

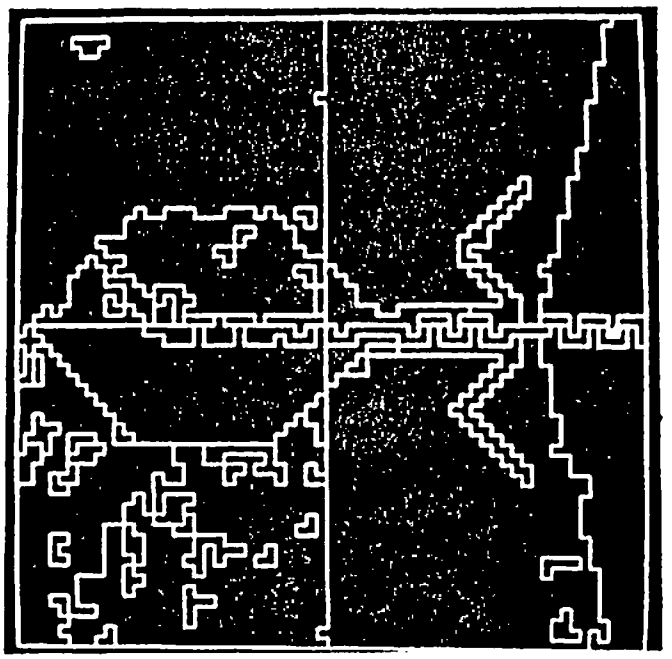


(a)

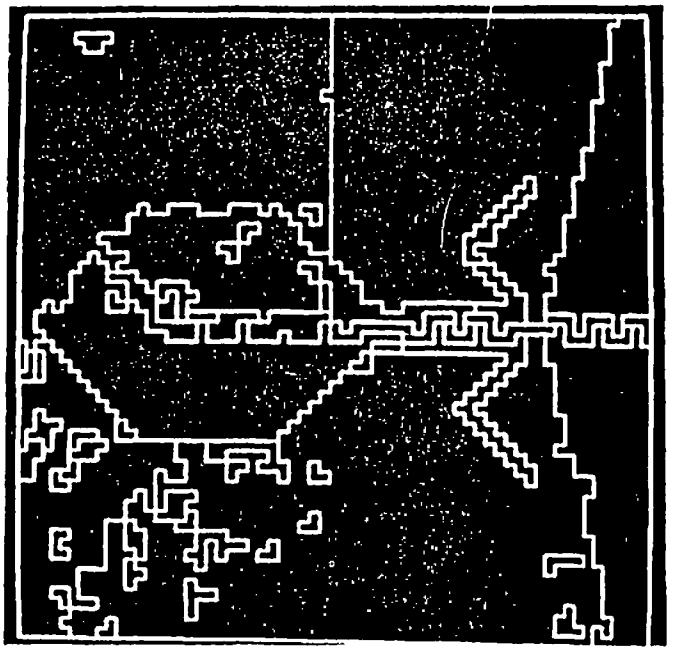


(b)

Figure 26: Virus - Coarse Localization - Without Cluster Addition.

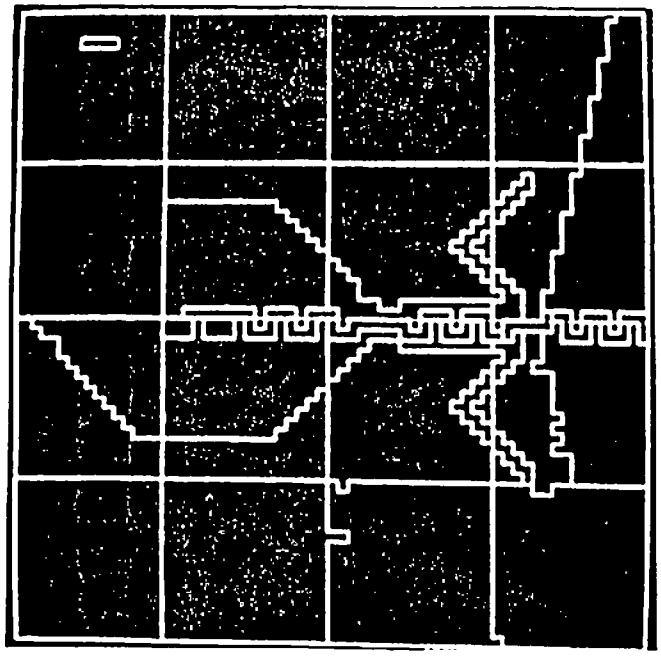


(a)

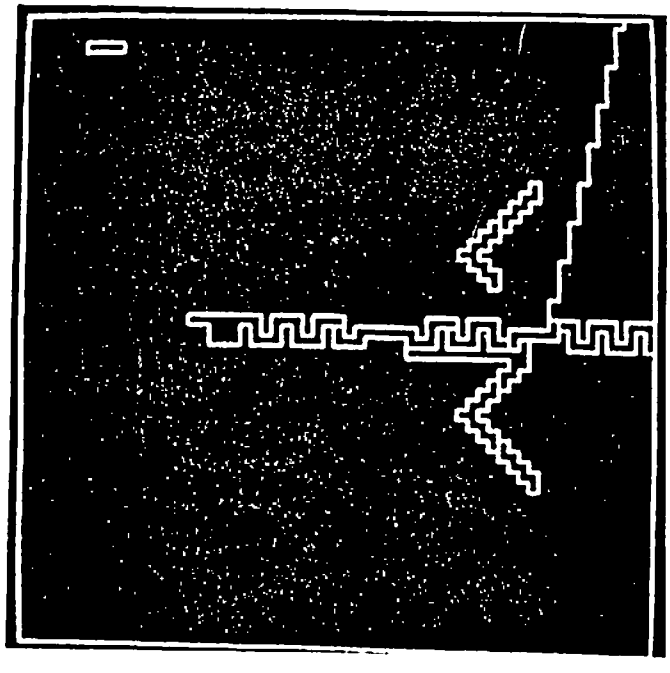


(b)

Figure 27: Virus - Coarse Localization - With Cluster Addition.



(a)



(b)

Figure 28: Virus - Fine Localization - Without Cluster Addition.

subimage boundary. The resulting segmentation after cluster addition is shown in figures 29a and 29b. Again, cluster addition located the missing clusters in the two subimages, producing a final segmentation with fewer gross errors.

Figure 30 shows that the same problems occur with images of natural scenes (in this case a house scene) and that the cluster addition algorithm is effective for this class of image as well. Figure 30a is the intensity of the house scene used in this example. Figure 30b is the segmentation of the intensity image obtained without the cluster addition algorithm and before artificial inter-subimage boundaries are removed. The circle in figure 30b highlights a striking example in which two clusters, representing the house wall and sky, were detected as a single cluster. As a result, the boundary separating these regions is missing. Figure 30c shows the corresponding segmentation after these boundaries are removed. Figure 30d and 30e show the corresponding segmentations when the cluster addition algorithm was incorporated into the segmentation process. Again, the cluster addition algorithm produced a qualitatively better segmentation and, in particular, separated the sky and house in the indicated subimage.

6.5 Cluster Deletion

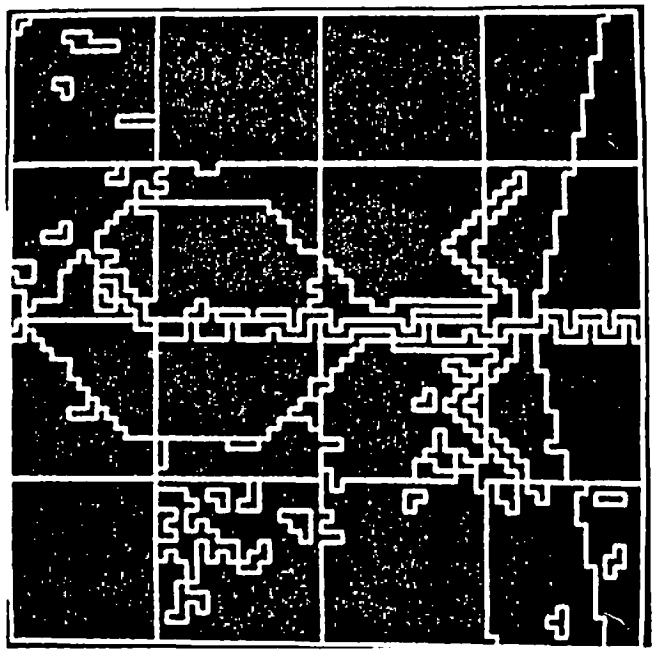
In this section we introduce several techniques which may be used to delete clusters based on spatial information not available in the feature histogram used by the cluster selection algorithm. The primary assumption of the region relaxation algorithm (and in fact, most segmentation algorithms based on clustering) is that a cluster in the feature space corresponds to a "useful" or "meaningful" region or set of regions in the image space. This assumption can be violated in a number of situations, including the following:

(1) The pixels which generated the cluster may not form reasonably large spatially contiguous regions since the clustering typically does not utilize spatial information. This could result in segmenting areas with microtexture into hundreds of separate small regions.

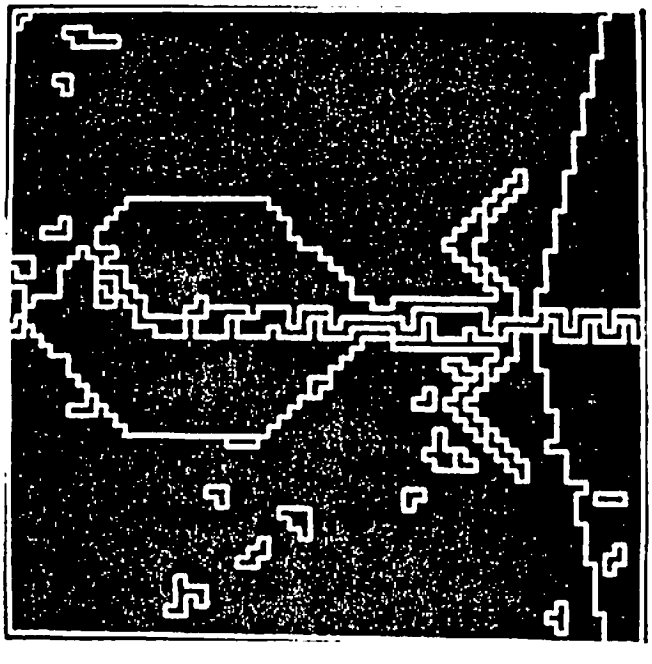
(2) The clusters may be due to a digitization or blurring artifact. A small cluster may be found between two significantly different clusters when the feature values of "mixed" pixels, which straddle the boundary between two image regions corresponding to the large clusters, have very similar feature values. Thus, the small cluster generally corresponds to a very narrow region between the regions corresponding to the large clusters.

(3) Multiple clusters may be found for a single region when the region exhibits spatially varying feature values (e.g. an intensity gradient). The extra clusters result in undesirable fragmentation of the region.

It is not possible to recognize these cases and delete the incorrect clusters based on the feature space information alone. The spatial distribution of the pixels associated with the cluster must be considered. The following sections propose several independent knowledge sources which validate or reject clusters by estimating



(a)



(b)

Figure 29: Virus - Fine Localization - With Cluster Addition.

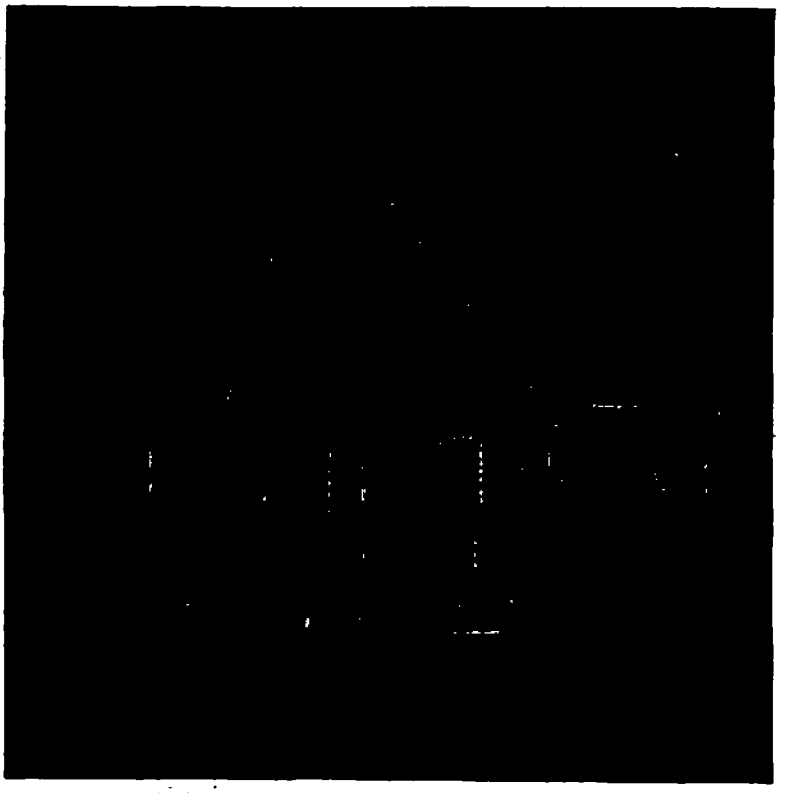
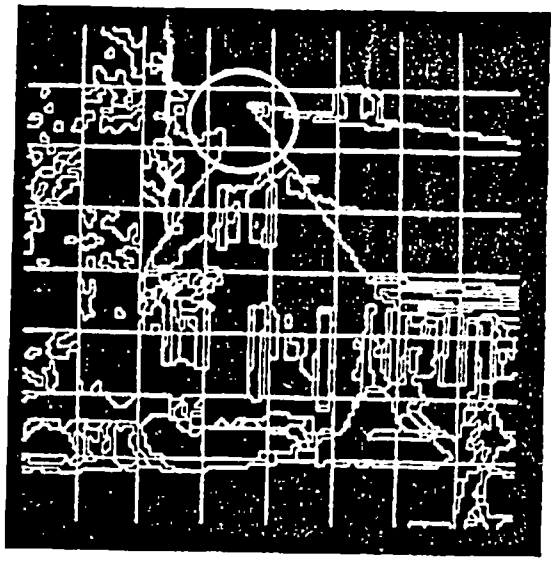
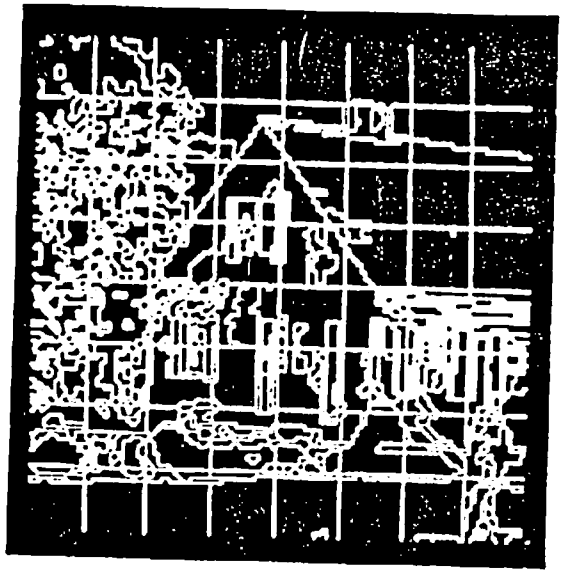


Figure 30: Cluster Addition for House Scene.



(b)

(c)



(d)

(e)

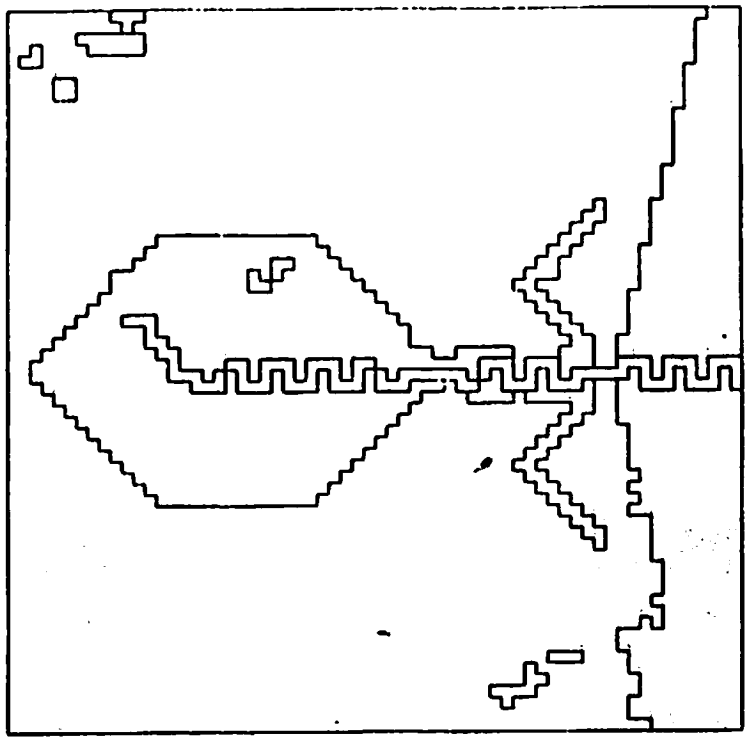
Figure 30: Continued.

the utility or contribution of that cluster in the final segmentation. Since the relaxation is computationally expensive, we do not want to complete the relaxation process to determine the spatial validity of a given cluster. Instead we would like to inexpensively estimate the segmentation expected if the hypothesised cluster set were adopted.

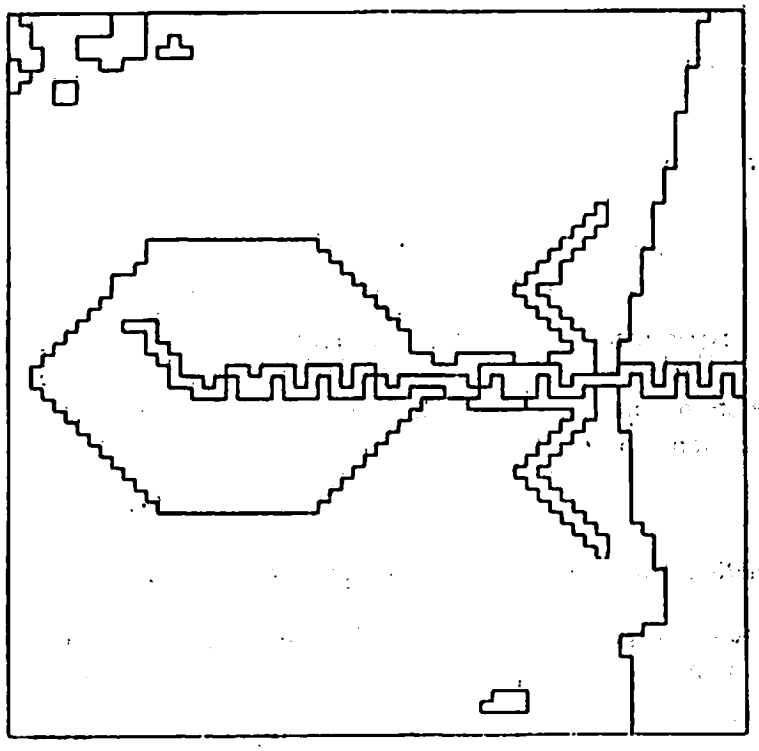
The cluster utility of any cluster is then evaluated relative to this estimate of the segmentation (utilizing the proposed cluster set). The estimate of the segmentation, utilized in the following sections, is computed by labeling each pixel with its initial best label and then allowing one and two pixel regions to be absorbed by their surrounding regions. This segmentation estimate is relatively inexpensive to compute (as compared to the iterative relaxation algorithm) and quite accurately predicts the gross structure of the final segmentation, as demonstrated by the following example. Figure 31 shows the estimated segmentation for a global segmentation of the virus image and the corresponding segmentation after forty iterations of relaxation. As can be seen, the estimate and the final segmentation agree quite well. One might even argue that given the striking similarity of the inexpensive estimate and the expensive relaxation-based segmentation, that the estimation algorithm by itself is adequate for many segmentation problems. We will return to discuss this issue in the conclusion of the dissertation.

Note that a feature cluster will often project onto several distinct image regions. Some of the regions formed by a cluster may be due to the effects listed earlier, while other regions may correctly correspond to several objects, object parts, or surfaces in the image. It is possible to extend the cluster deletion methodology to each region, utilizing the same kind of information to make decisions separately for each region, after the relaxation process. This approach is discussed in section 6.6.2 below. It is certainly desirable to delete clusters which do not correspond to "meaningful" regions in the scene before relaxation (rather than relying totally on the region merging) for practical reasons. Relaxation is, by far, the most expensive component of the segmentation process, and the cost of the relaxation algorithm is proportional to the square of the number of clusters. In addition, the number of iterations of relaxation required to reach local convergence has been observed to increase with the number of labels since each additional label at a pixel implies higher initial ambiguity as to the correct labeling of that pixel.

Another observation about the cluster deletion processes which we are about to discuss is that they are essentially an error correction mechanism for the cluster selection process. However, each of the cluster deletion operators will typically find very few corrections to make in any given image and, for most subimages, no clusters will be deleted. Thus, the cost of computing the cluster deletion tests must be weighed against the possible benefits of correcting the cluster set using that particular operator. The cluster deletion tests developed were kept as computationally inexpensive as possible while still finding poor clusters. Since cluster deletion is a relatively rare event in general, we have



(a)



(b)

Figure 31: Estimation of Region Relaxation Segmentations.

intentionally selected too many clusters in the experiments presented below in order to effectively evaluate the deletion operators.

Cluster deletion using region size.

If a cluster does not form any fairly large regions in the image, that cluster may be due to image noise, highlights or micro-texture. Typically, regions which can be attributed to any of these causes are not desirable in the segmentation, although this is of course a function of the goal of the segmentation process. The minimum size of regions to be found by the segmentation is clearly a function of the image domain. Regions of less than three or four pixels are probably not meaningful regions in almost all domains.

By performing a connected component analysis (i.e. region labeling), one can determine the maximum size, average size, and size variance for all regions formed from a given cluster. If the size of the largest region is less than some conservative threshold, then the cluster will be deleted from the cluster set. A cluster which forms a large number of small regions which have little variance in size is typically indicative of a distributed micro-texture. Although this information may be valuable in later processing, we do not believe that it is generally desirable to produce initial segmentations with large numbers of regions corresponding to texture elements.

This particular cluster deletion operator was not used in our system for a number of reasons:

(1) connected component analysis is a fairly expensive operation for a large image with many regions;

(2) empirical experience indicated that very few incorrect clusters could be detected with reasonably conservative threshold sizes; and

(3) empirical experience indicated that clusters which would have been deleted by the size operator would generally be deleted by either the mixed-pixel cluster deletion operator (section 6.5.2) or the compactness constraint operator (section 6.5.3).

Cluster deletion for mixed-pixel or blurred boundaries.

If image boundaries do not coincide exactly with the digitization grid (as shown in figure 32b), pixels with intermediate values will be generated at an object boundary (see figure 32c). Effects due to a discrete representation of continuous image events are called aliasing effects in computer graphics. While "mixed" pixels are desirable in computer graphics because they tend to produce a more visually smooth edge, in image analysis they may form observable clusters in the histogram space (as shown by figure 32d) leading to incorrect segmentation (as shown in figure

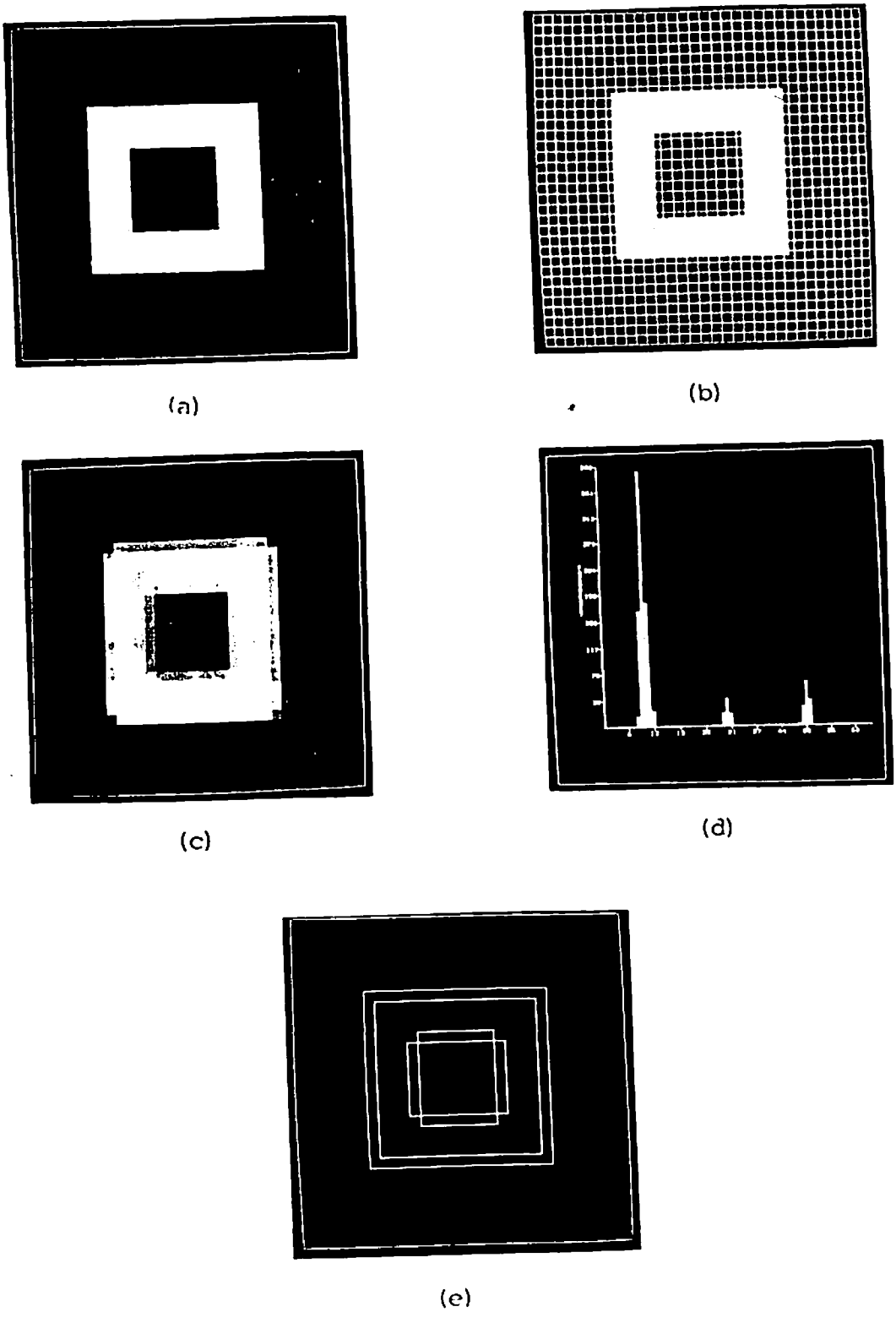


Figure 32: Clusters Due to Digitization Errors.

32e). It is desirable to detect such clusters and delete them from the cluster set.[15]

There are image characteristics which can effectively discriminate these kinds of clusters from desirable clusters. In particular, the regions formed by the mixed pixel cluster tend to be narrow and each pixel in one of these clusters will be adjacent to at least one pixel of a darker cluster (smaller mean) and one pixel of a brighter cluster (larger mean). A simple statistic which estimates these properties is defined as follows. Let B_i be the spatial distance, from pixel i, to the nearest pixel with a brighter cluster label and let D_i be the spatial distance to the nearest dimmer cluster label. We define a measure at each pixel to be

$$M(i) = \frac{2}{B_i + D_i} \qquad 6.1$$

This measure is 1 when pixel i lies between and adjacent to a brighter and darker region and approaches 0 when i is in the interior of a large region. The operator considered any D_i or $B_i > 2$ to be at infinity since we assume that no gradients wider than this should be interpreted as blurred boundaries or mixed pixels. The statistic for cluster C (of n pixels) is defined as the average score at each pixel:

$$S(C) = \frac{\sum_{i \in C} M(i)}{n} \qquad 6.2$$

This measure will be close to 1 for clusters which should be deleted and less than 1 for clusters which should be kept.

Once a cluster due to this effect is discovered, the cluster is deleted from the set of clusters. The pixels in the cluster should be labeled with either the dimmer or brighter neighboring cluster based on the pixel's feature values and the local image context of the pixel. Deletion of the cluster automatically has the effect of collapsing the decision boundaries that separated the deleted cluster and its dimmer and brighter neighboring cluster into a single decision boundary at the deleted cluster. This results in very ambiguous initial cluster affiliation probabilities at pixels within the deleted clusters. The relaxation process can then use local spatial image context to make the final decision for each pixel in the deleted cluster.

Figure 33a shows a subimage of the house seen in previous figures. The initial segmentation is shown in figure 33b. Cluster are deleted if the mixed pixel score is close to 1.0 ($t > .75$ is considered close to 1.0 in this example). Figure

[15] It is also possible to attack this problem directly in the image space by locating the mixed pixels and modifying them to match their closest neighbors in a pre-processing algorithm. This approach was effectively used in color images by Prager [PRA79]; it is much less reliable in monochromatic images.

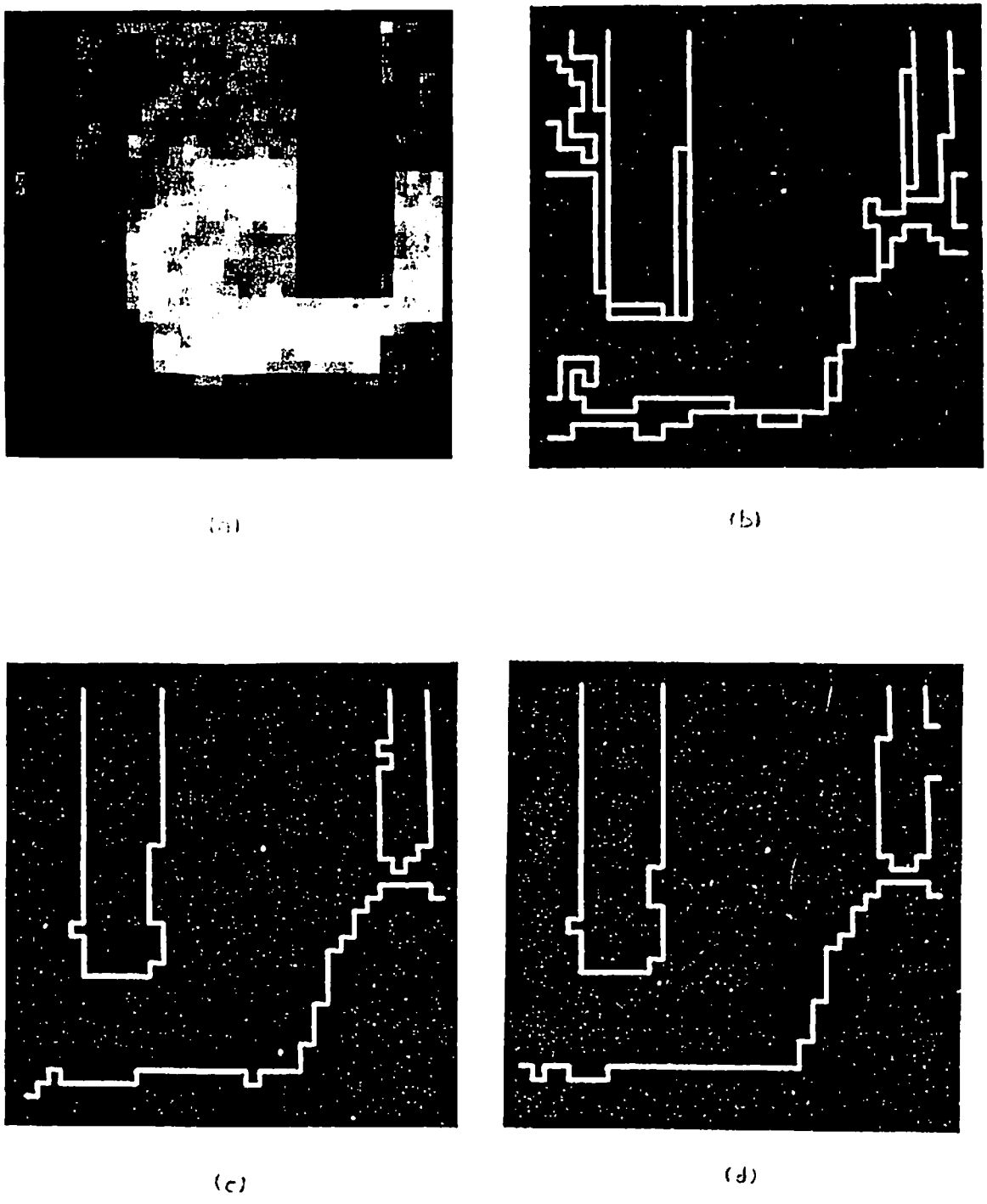


Figure 33: Mixed Pixel Cluster Deletion Example.

33c shows the initial labeling after this cluster is deleted, while figure 33d shows the final segmentation after 40 iterations of relaxation.

A problem occurs when two or more adjacent clusters are found to be "mixed" pixel clusters. These relatively rare events are typically not due to digitization artifacts but to fairly sharp intensity gradients in the image. The region with the intensity gradient leads to several distinct clusters in the histogram and is therefore fragmented into contour bands and the "mixed" pixel cluster score for each of these clusters may be quite high. These sets of adjacent high score clusters could be handled in several different ways:

(1) they can be left to exist as separate clusters,

(2) they can be merged into a single separate cluster,

(3) they can be split between the low score clusters above and below.

The best choice for dealing with these clusters is not clear and seems to be dependent on the characteristics of the image and the goals of the interpretation algorithm. The current implementation takes the most conservative path of allowing all of the adjacent gradient clusters to remain. The information that these clusters are not particularly desirable could be maintained so that, later, the application of other cluster deletion operators could favor merges incorporating these clusters or that processes specific to gradient regions could be invoked over the resulting regions.

Cluster Deletion using Spatial Compactness.

It is possible to find some undesirable clusters using a spatial compactness measure computed across the regions produced by a cluster. Clusters due to micro-texture and clusters in a sharp intensity gradient will often result in regions with low compactness scores. However, some clusters corresponding to fine structures in the image would also result in low compactness scores. A heuristic algorithm was devised to locate those clusters which should be merged with adjacent clusters based on the compactness score of the regions corresponding to the cluster.

The compactness measure for cluster i was defined to be:

$$C(i) = \frac{\text{Number of adjacencies between pixels in cluster i}}{2 * \text{Number of Pixels in cluster i}} \qquad 6.3$$

For a cluster mapping to a set of one pixel micro-texture regions this score would be zero since cluster i pixels are never adjacent to other cluster i pixels. For a region consisting of an infinitely long row one pixel wide the score would be one half since there are as many adjacencies as pixels. For a large globular region the score would be close to one since the average pixel has four adjacencies shared by

the four neighbors or an average of two adjacencies per pixel. The measure is bounded between zero and one and provides an intuitively reasonable compactness score for clusters composed of multiple regions. In addition, the measure is easily computed for a hypothetical merge of a set of clusters. Cluster merges are proposed only for clusters with very low compactness scores for which the compactness measure of the merged cluster is significantly higher than either initial cluster.

In some cases, the clusters to be merged are separated by a strong boundary. If this is the case, merging the clusters might eliminate some small image structures. Therefore, we block the merging of clusters if the regions corresponding to those clusters are seperated by relatively high contrast boundaries.

The contrast measure used to block potential merges is a normalized measure of the difference in edge strength within each of the two clusters which might be merged and the edge strength along the boundary between the two clusters. The measure is detailed in section 6.5.4

The virus image (Figure 24) is used here to demonstrate cluster deletion based on the compactness constraint. The cluster selection mechanism was adjusted so as to select even weak clusters. For the virus image, this resulted in nine clusters where four is known to be optimal. Figure 34a shows the initial segmentation (without relaxation) produced by these nine clusters. Figure 34b shows the segmentation after the compactness based cluster deletion algorithm. Figure 34c shows the same result with one and two pixel regions suppressed. Figure 34d shows the segmentation after forty iterations of relaxation. All of the five superfluous clusters were correctly deleted by this operator. In general, this algorithm will only delete those clusters which correspond to micro-texture or to intensity gradients. In this example, it happened that all of the clusters deleted fell into these types.

Cluster Deletion using Edge Information.

As described earlier, wide intensity gradients can often lead to multiple clusters. The goal of the processing proposed in this section is to locate adjacent clusters which correspond to continuously and slowly varying feature changes over possibly wide areas of the image. Thus, the assumption that there are no abrupt changes (i.e. large magnitude in the gradient) across the intensity gradient allows one to expect very weak edges (intensity gradients) between image regions corresponding to a single continuous intensity gradient; this contrasts with the edge measures between regions belonging to different image structures. In short, multiple clusters from the same gradient should not have edge information in the image to support the separation of those clusters. The method proposed for detecting such clusters attempts to bring edge expectancies into the cluster selection process. Once detected, these clusters which are not separated by strong edges could be merged. Naturally, this cluster evaluation only makes sense when the clusters to be merged form at least some adjacent regions in the image.

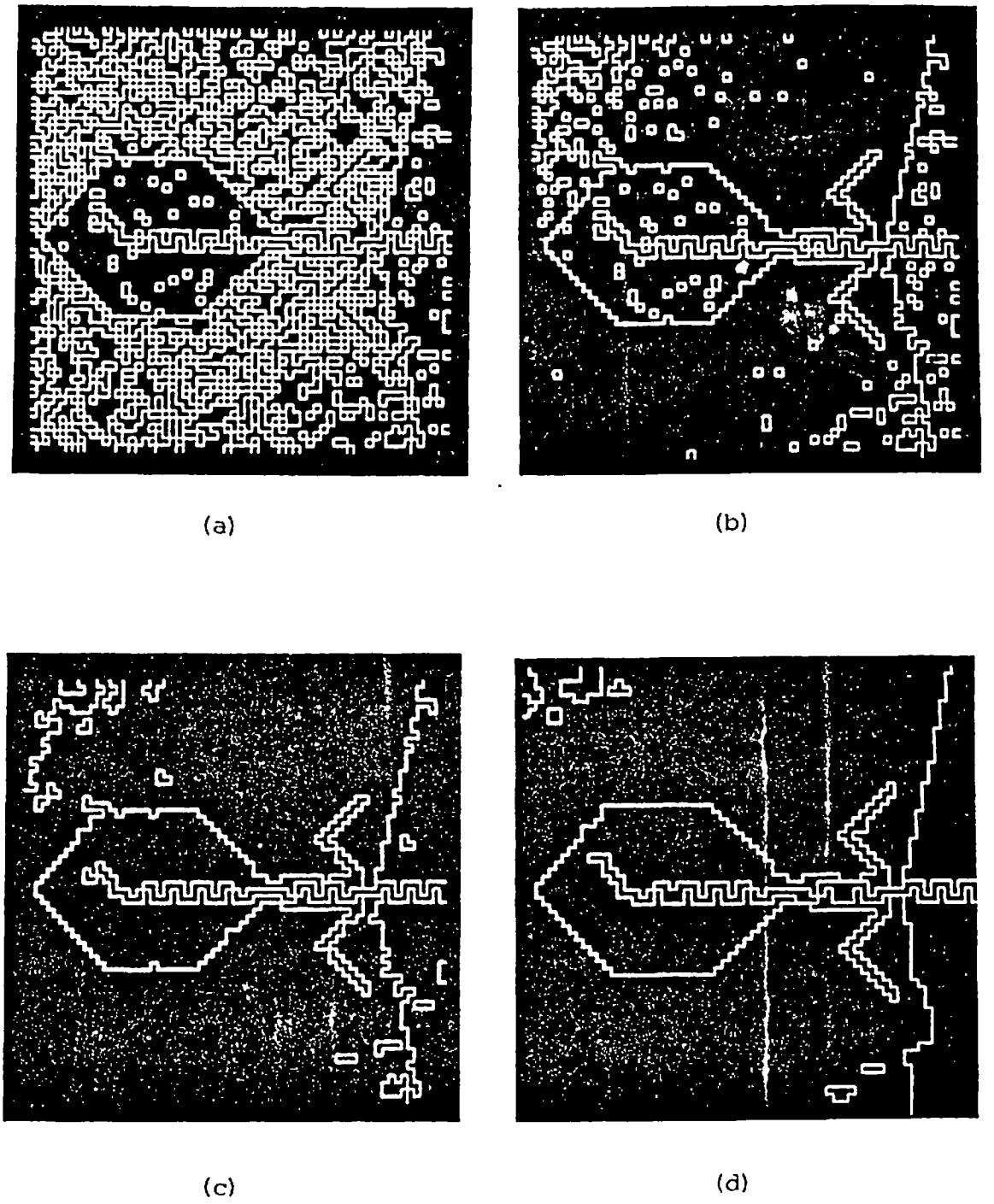


Figure 34: Cluster Deletion using Compactness for the Virus Example.

Any measure of edge strength between regions formed by adjacent clusters should be normalized by the average edge strength found in the interior of these regions. The actual contrast measurements considered were based on the raw horizontal and vertical inter-pixel differences and the edge probabilities produced by the edge relaxation algorithm after a varying number of iterations.

Several inter-cluster contrast measures were considered for comparing the contrast measure over the population of edges between pixels interior to cluster i (denoted E_{ii}), the set of edges interior to cluster j (denoted E_{jj}), and along the boundaries which separate regions formed by clusters i and j (denoted E_{ij}).

The first measure considered was based on the Neyman-Pearson statistical test as proposed by Yakimovsky [YAK73]. Given the size of E_{ii} is k edges and the size of E_{ij} of is m edges and the size of E_{jj} is n edges, the measure is defined as:

$$\text{Max}\left(\frac{\sigma(E_{ii} \cup E_{ij})^{k+m}}{\sigma(E_{ii})^{k}+\sigma(E_{ij})^{m}} , \frac{\sigma(E_{jj} \cup E_{ij})^{n+m}}{\sigma(E_{jj})^{n}+\sigma(E_{ij})^{m}} , \frac{\sigma(E_{ii} \cup E_{jj})^{k+n}}{\sigma(E_{ii})^{k}+\sigma(E_{jj})^{n}} \right). \qquad 6.4$$

The measure is small when two populations have identical distributions and large when the two populations differ either in their mean or variance. The terms in this measure correspond to the Yakimovsky measure applied to the following three populations:

(1) E_{ii} and E_{ij},
(2) E_{jj} and E_{ij},
(3) and E_{ii} and E_{jj}.

If the edge population associated with regions in either cluster is significantly weaker than the edge contrast population between the clusters, or if the the edge contrast populations within the two clusters differ significantly then this measure will be large.

Unfortunately, this measure was not found to be reliable in consistently locating clusters lying within a single intensity gradient. In particular, when the regions were large, the statistic could detect differences in the mean and variance of the populations which were not visually discernable by a human, while the measure did not always detect population differences when the regions were small and visually discernable. Furthermore, if the E_{ij} term were smaller than both the E_{ii} and E_{jj} terms, indicating that no edge was present between the clusters, the measure still produces a high score since the populations are indeed different.

Another measure considered was a modification of the Yakimovsky measure proposed by Nagin [Nag79]. Although somewhat more reliable, the measure was also ineffective in consistently locating the clusters which lay within a single intensity gradient. This measure was not sensitive to the cluster sizes, but also failed when

E_{ij} was smaller than E_{ii} and E_{jj}.

The most reliable measure found for merging clusters which lay within a single intensity gradient was a simple normalized difference of the means of the contrast measure populations:

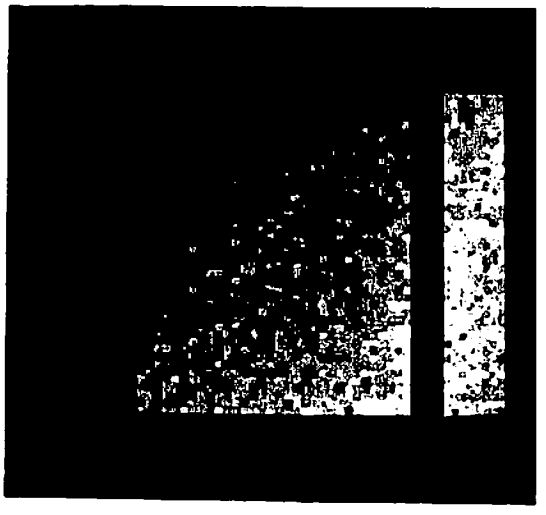
$$M = \text{Max}\left(\frac{\mu(E_{ij}) - \mu(E_{ii})}{\sigma(E_{ij}) + \sigma(E_{ii})}, \frac{\mu(E_{ij}) - \mu(E_{jj})}{\sigma(E_{ij}) + \sigma(E_{jj})}, \frac{\mu(E_{ii}) - \mu(E_{jj})}{\sigma(E_{ii}) + \sigma(E_{jj})} \right) \quad 6.5$$

Figure 35 shows an example in which the measure is effective. The central rectangle of figure 35a has a wide slowly varying intensity gradient across it. Figure 35b shows the histogram computed from 35a. Rather than two or even three distinct peaks corresponding to the foreground rectangles the histogram is a noisy blur between the clusters corresponding to the left and right rectangles. Figure 35c shows the estimated segmentation based on the five clusters chosen from this noisy histogram. Table 1 shows the edge measures for each pair of adjacent clusters. As can be seen, the average edge strength between cluster 1 and cluster 2 is very much larger than the edge strength between any other pair of adjacent clusters. Figure 35d shows the segmentation after all cluster pairs with low edge scores (clusters 2 and 3, 3 and 4, and 4 and 5) are merged.

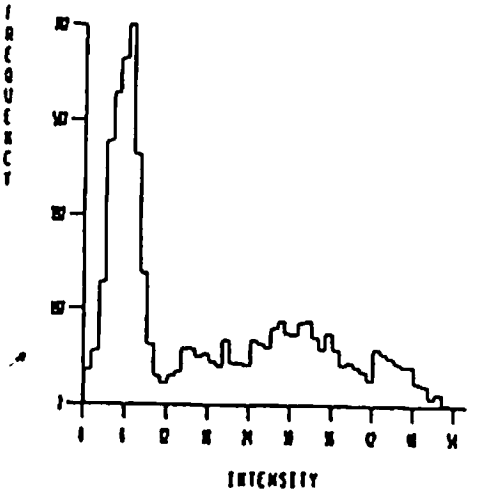
The basic premise of this cluster deletion approach is that there will always be significant edges between regions of interest in the image. This assumption is not always valid, as demonstrated by the virus example of figure 24. In the lower right corner of the virus image there is little or no edge information to support the boundary between the background and the E. Coli cell. It was only through the inter-subimage cluster addition algorithm that we were able to distinguish the background and E. Coli cell at all.[16]

Although the gradient cluster detection mechanism is fairly robust, at this stage in the algorithm's development, non-gradient clusters are occasionally deleted. Deletion of a non-gradient cluster can of course result in significant undersegmentation, however if gradient clusters are allowed to remain the gradient region will typically be split into several fairly large regions producing a slight oversegmentation. One avenue that is still being explored for merging these clusters (or regions) is based on a fit of a planar surface to each cluster (or region). Regions or clusters could, of course, be modeled by more sophisticated surface representations as well (e.g. bicubic splines) [HAR79]. If adjacent clusters have very similar best fit surfaces then the clusters (or regions) should probably be merged.

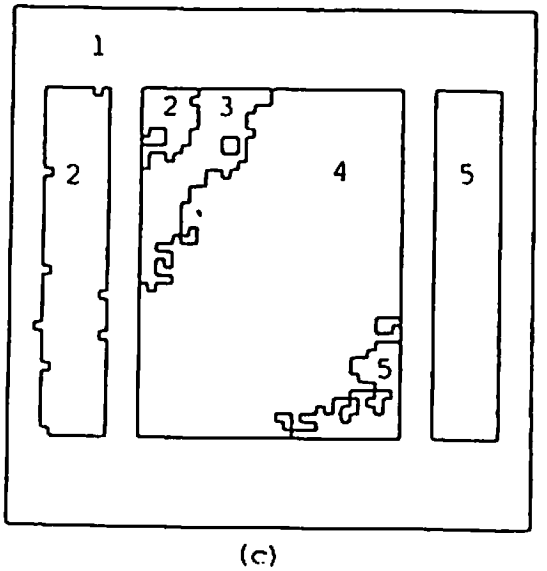
[16] Note that the boundary is almost implied in such cases. Humans use a variety of mechanisms including subjective contour mechanisms and high level models to locate such boundaries. A slowly changing intensity gradient may represent a boundary between objects or a slight change in surface orientation of a curved surface. Distinguishing these cases without precise goals and world knowledge may well be impossible.



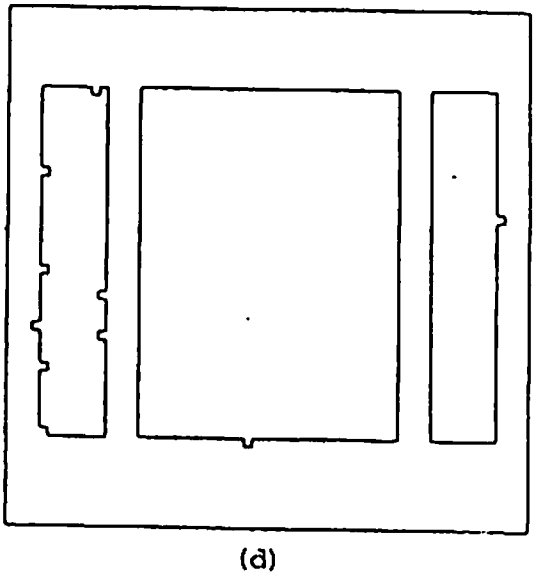
(a)



(b)



(c)



(d)

Figure 35: Merging Clusters in an Intensity Gradient.

Average Strength of Edges Between Pixels Labeled i and j

(j) \ (i)	1	2	3	4	5
1	3892				
2	132	693			
3	24	38	249		
4	84	8	72	2858	
5	128	8	8	76	754

Average Strength of Edges Between Pixels Labeled i and j

(j) \ (i)	1	2	3	4	5
1	.12				
2	.95	.25			
3	1.00	.21	.04		
4	1.00	.00	.22	.17	
5	1.00	.00	.00	.24	.00

Variance of Edge Strengths Between Pixels Labeled i and j

(j) \ (i)	1	2	3	4	5
1	.09				
2	.04	.16			
3	.00	.15	.04		
4	.00	.00	.14	.12	
5	.00	.00	.00	.16	.07

Table 1: Inter-Cluster Edge Co-occurrence Matrices.

6.6 Region Merging

The relaxation process produces separate segmentations for each of the localized subimages. In order to obtain a single segmentation for the entire image, it is necessary to combine regions in adjacent subimages if they are part of the same image structure. Beyond this particular need, some of the criteria used to merge or delete clusters may be applied to merge or delete regions and thereby improve the segmentation by reducing oversegmentation. A decision to delete a cluster has an impact on all of the regions which belong to the cluster; by making the same types of decisions made for clusters on a region by region basis we can utilize the same knowledge to make more localized decisions. Many of the issues shown to be important for cluster deletion are incorporated into the region merging mechanism developed here. By making the merge decisions at the level of regions we actually obviate the need for the cluster deletion.

Merging Regions Across Sub-Image Boundaries.

Nagin [NAG79] merged adjacent regions across subimage boundaries when a statistical test (similar to the measure used by Yakimovsky [YAK73] and discussed in section 6.5.4) indicated that small region areas on either side of the boundary could have come from the same population. Note that the merging criterion was based solely on a local measure of the candidate regions near the subimage boundary.

In the ideal case a threshold is selected for this measure such that all artificial boundaries are deleted, but no correct image boundaries are deleted. The failure to merge two regions which correspond to the same semantic structure might lead to oversegmentation. On the other hand, a single incorrect merge could have devastating effects, causing the indirect merging of several large distinct structures. Figure 28 is a case where an undesirable merge occurred. Figure 28a shows the sixteen subimage segmentations before remerging, while figure 28b shows the segmentation after remerging. The poor quality of the final segmentation is due to two incorrect merges through the subimages which had missing clusters. The remerging problem is particularly difficult when one of the subimages is missing a correct cluster. In this case there is little or no local support along the subimage boundary for the proposition that the two regions should not be merged as shown in figure 36. In this figure the right subimage containing R3 is presumed to be missing a cluster; thus a merge of R1 with R3 and R2 with R3 will have the undesired effect of indirectly merging R1 and R2 and losing the boundary between these regions in the segmentation. In this example both merges are locally correct, and the merge will propagate the undesirable effect of the missing cluster in the right subimage to the left subimage and perhaps indirectly to other subimages.

Figure 37 shows a remerging example with some possible outcomes. Figure 37a shows an image divided into four subimages. Figure 37b shows the regions produced by the localized segmentation process. Figure 37c shows the ideal segmentation in which all merges have been correctly performed while figure 37d

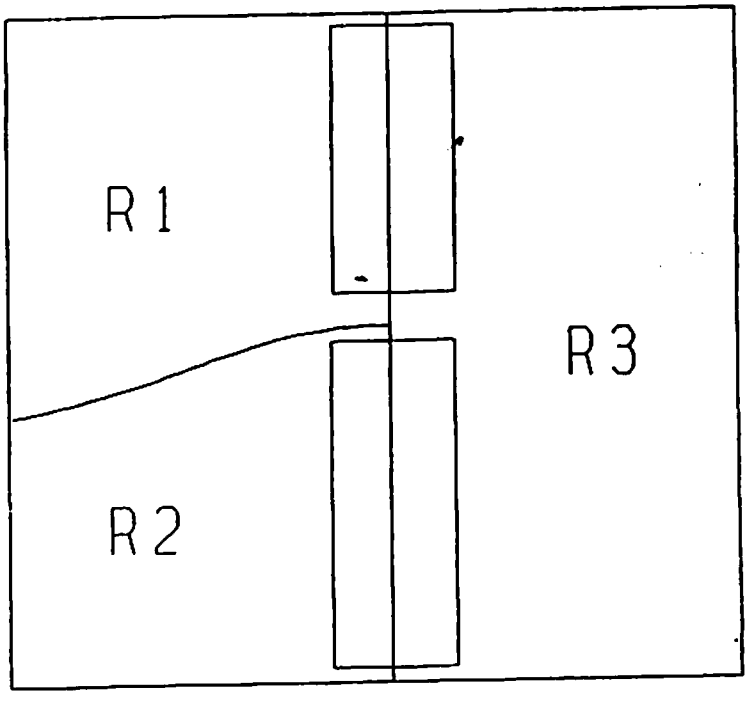


Figure 36: A Missing Cluster Merging Example.

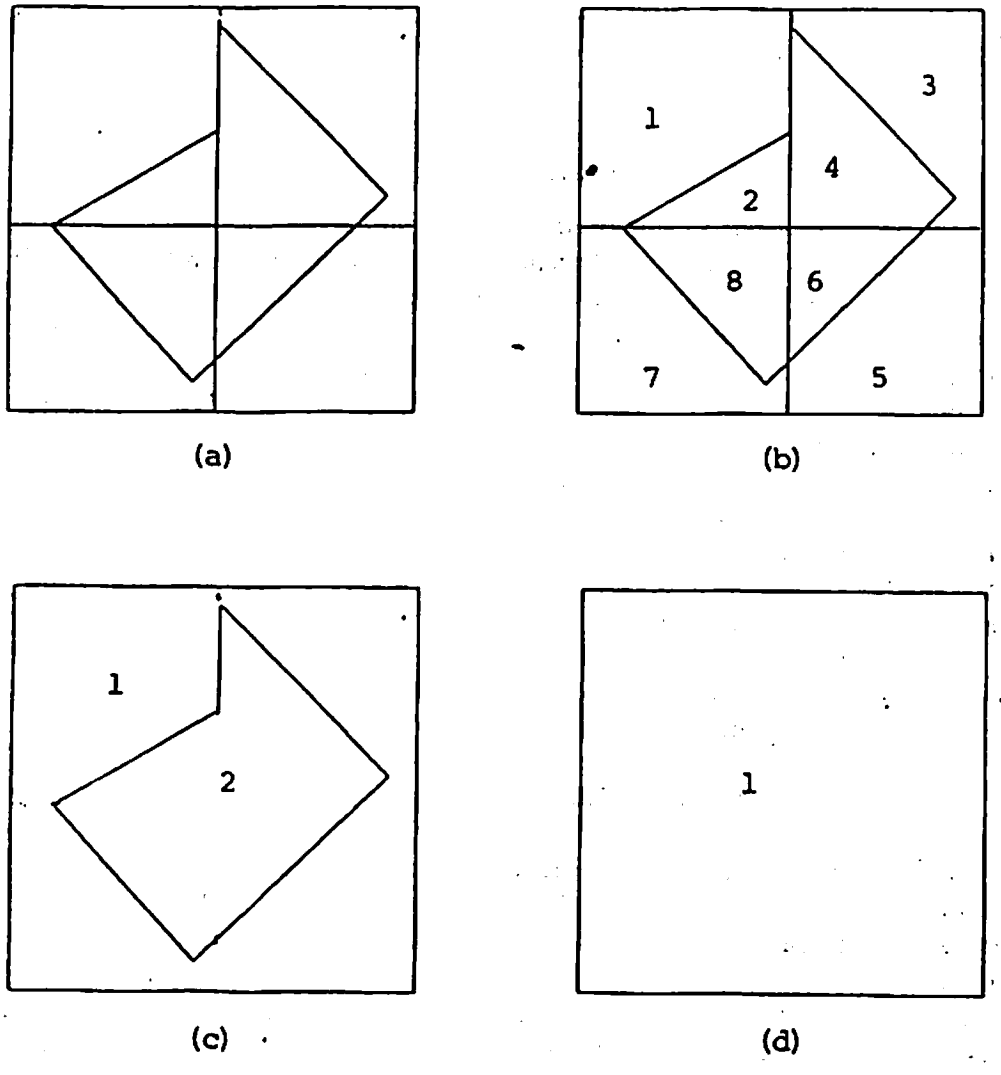


Figure 37: Prototypical Subimage Merging Example.

shows the effect of not merging regions two and eight and regions four and six. Finally figure 37e shows the effect of the single incorrect merge joining regions one and four.

Although the merge criterion of locally identical population statistics used by Nagin is quite reliable, it is, in general, impossible to select a threshold which performs all desirable region merges without making any undesirable merges. In order to improve the overall performance of the segmentation algorithm two steps were taken:

(1) the merging criterion was extended to include a more global region measure

(2) and a conservative merging threshold was selected.

The merging criteria employed here permits a merge between two regions in different subimages which are adjacent at a sub-image boundary only if both a local constraint and a global constraint are satisfied. The local constraint is identical to the constraint used by Nagin. This constraint is satisfied when the population similarity measure between the small areas on either side of the boundary is below a threshold. The global constraint is satisfied when the same measure computed across the complete regions is below another threshold. By using a local and a global threshold one can use a more liberal local threshold, which alone would allow some undesirable merges to occur. The global threshold will block those merges which would be incorrect. In the case of missing clusters (see figure 36), both possible merges of R1 with R3 and R2 with R3 would satisfy a local constraint, but neither R1 nor R2 should be able to merge with R3 based on the global constraint. Note that using a global merging criterion alone would not be as effective since fairly similar regions with a high contrast edge at the boundary should also not be merged. In this case the local constraint would block the merge.

Region Merging Based on Other Constraints.

Even after careful cluster selection, relaxation, and merging across sub-image boundaries, the segmentation typically contains some adjacent regions which correspond to the same image entity and which should be merged. In the algorithm presented below a number of different knowledge sources, using different information, interact to select desirable region merges. Each knowledge source embeds some heuristic strategy as to whether a region should be merged with any of its neighbors, and if so, which of its neighbors it should be merged with. Although the actual knowledge sources used below are very simple, more complex knowledge sources and varied merging knowledge sources are possible. Specialized knowledge sources which detect very particular image events can be designed and one could even utilize feedback from the semantic interpretation system in making merge decisions.

The algorithm for region merging is abstracted in figure 38. The first phase of the merging algorithm generates a set of merge hypotheses: a set of regions which are to be considered for merging with one of their neighboring regions. The

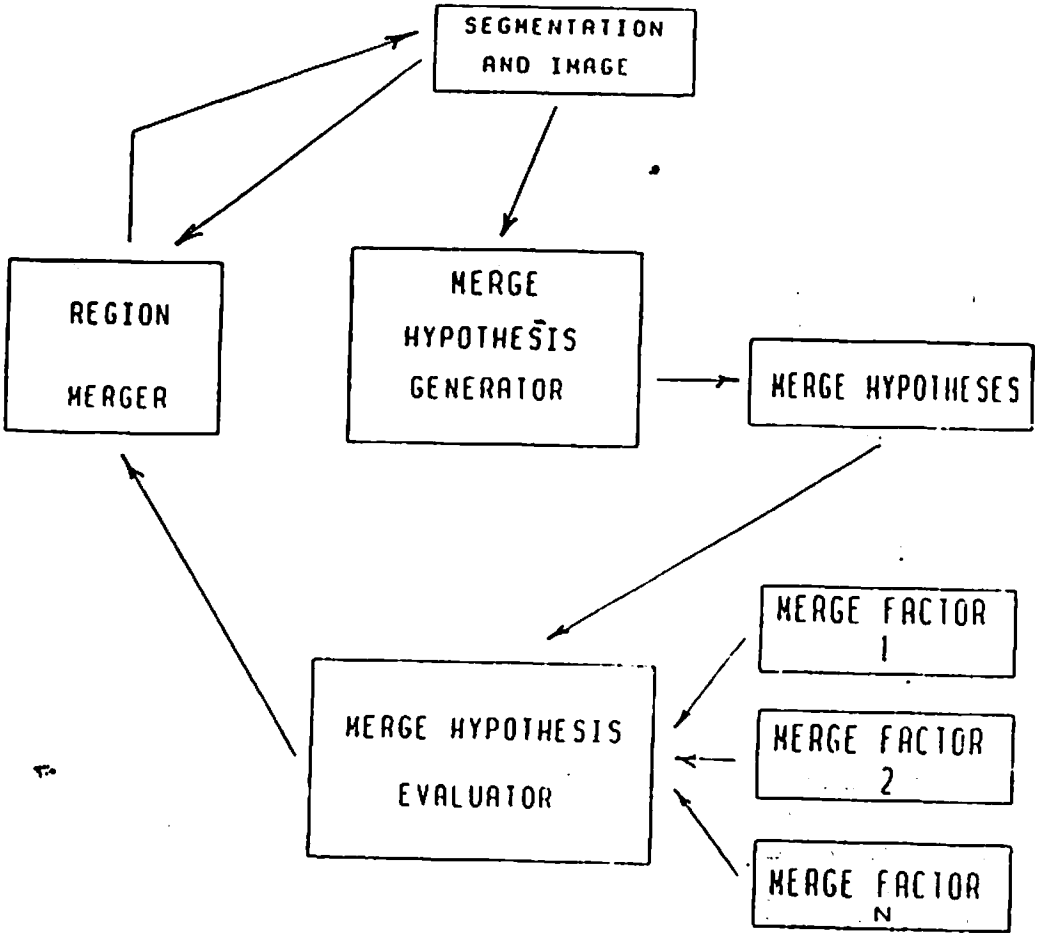


Figure 38: Region Merging Algorithm - Overview.

second phase evaluates the region merge hypotheses by applying a set of knowledge sources (described below) to each merge candidate region. For each candidate, the merge with each of its neighbors is considered. The knowledge sources applied in this phase influence a merge score which represents the efficacy of each possible merge of the candidate.[17] The third, and last, phase filters the merge hypotheses by merging a candidate with its neighbor with the best merge score if that merge score is sufficiently high. These phases may then be iterated until no more merges take place.

Generating Merge Hypotheses:

It is possible to simply allow all regions to be candidates for merges and count on the filtering process to reject the undesirable merges. This would entail considerable computation in determining merge scores for all region adjacencies in the image. In order to reduce this cost, knowledge sources could add regions to be considered for merging to the set of merge candidates based on some characteristics of the regions or external input such as feedback from a semantic interpretation process.

In this thesis no such knowledge source was utilized: all regions were made merge candidates.

Evaluating Merge Hypotheses:

For a given merge candidate region it is necessary to determine:

(1) whether the candidate region should merge with any of its neighbors and

(2) which of the neighbors corresponds to the "best" merge.

Both of these decisions require some evaluation function which will determine the similarity of two regions. The evaluation function is defined as a product of terms, where each term is independently computed by separate knowledge sources. The contribution of each knowledge source will be defined to be large (greater than 1.0) when two regions being compared are different (in the characteristic feature measured by that knowledge source) and small (in the interval 0.0 to 1.0) when the regions are very similar. When a given knowledge source cannot contribute any information then the merge factor computed should be 1.0. The global merge score, computed as the product of all of the merge factors, is then greater than 1.0 when the merge should not occur and less than 1.0 if the merge is reasonable. The algorithm will merge the candidate region with the neighbor with the lowest score if that score is less than 1.0, otherwise, no merge is performed.

---

[17] The current implementation utilized a simple multiplicative update rule, where each knowledge source simply multiplies the accumulated merge score by some evaluation computed by the knowledge source.

This merge evaluation function has a number of desirable characteristics:

(1) An arbitrary number of knowledge sources could contribute to the decision.

(2) The order of knowledge source evaluation has no effect. This implies that with proper hardware all of the knowledge sources could be computed simultaneously.

(3) The effect or contribution of each knowledge source can be separately tuned. One can dynamically enable or disable a knowledge source. It should even be possible to limit or enhance any one of the knowledge sources dynamically using knowledge about the particular image domain or even using the current partial model developed by the model building component of the image interpretation system.

(4) It is possible for a single knowledge source to effectively veto a given merge or to force a candidate to merge with one of its neighbors.

The region merging algorithm was initially implemented using only four knowledge sources based on the following assumptions:

(1) Global Difference of Region Means. The regions to be merged should have roughly the same mean of feature values.

(2) Global Difference of Region Variances. The regions to be merged should have roughly the same variance in feature values.

(3) Minimum Candidate Region Size Constraint. Very small regions should almost always merge with one of their neighbors.

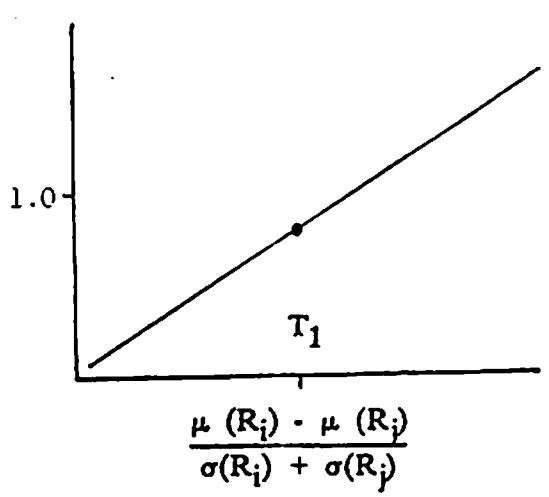
(4) Minimum Region Connectivity Constraint. It is desirable that regions to be merged share at least a minimum length of common boundary.

Each of these knowledge sources is discussed briefly below.

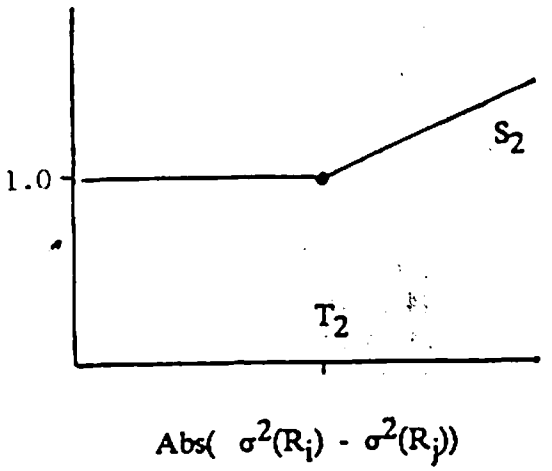
The intensity similarity of region means is probably the most important factor in deciding whether the regions should be merged. This knowledge source simply returns the scaled and normalized difference in the means of the two regions. For regions R_i and R_j the measure is defined as:

$$M_1 = \left(\frac{1.0}{T_1}\right) \frac{\mu(R_i) - \mu(R_j)}{\sigma(R_i) + \sigma(R_j)} \qquad 6.6$$

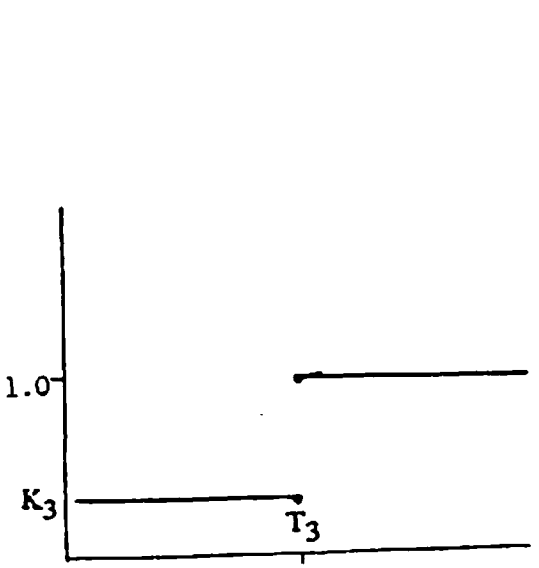
where T_1 is the threshold on the normalized difference. When the normalized difference of means is less than T_1, then M_1 is less than one, favoring the merge of the regions. The implementation actually includes a lower-bound on M_1 such that M_1 will not force a merge when there are other counter-indicators. The graph of figure 39a shows the merge factor (M_1) returned by this knowledge source as a



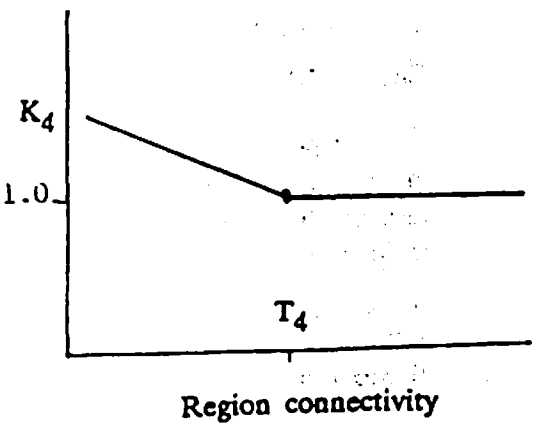
(a) Merge Factor M1



(b) Merge Factor M2



(c) Merge Factor M3



(d) Merge Factor M4

Figure 39: Merge Knowledge Sources - Merge Factor Graphs.

function of the measurement of the normalized difference of means.

It can be argued that regions with very different variances should probably not be merged, as in the case of regions with easily discernable texture differences, but the same average intensity. This second knowledge source serves to inhibit merges between regions of very different variances. The merge factor (M_2) returned by this knowledge source is shown in figure 39b. As can be seen, the factor is neutral (returns a value of 1.0) unless $Abs(\sigma^2(R_i) - \sigma^2(R_j)) > T_2$ in which case the operator tends to block the merge. The global effect of M_2 on the merge score is controlled by the magnitude of the slope S_2.

The third knowledge source serves to enforce the merge of very small regions. The assumption is that very small regions may be due to texture or sensor noise and should in general be deleted. This knowledge source serves to enforce the merge of any very small candidate regions. The merge factor (M_3) returned by this knowledge source is shown in figure 39c. This factor is based solely on the candidate region, and therefore effects all merge scores with its neighbors equally. This operator only effects the merge/don't merge decision since it does not affect the relative merit of merging the candidate with its neighbors. For candidate regions larger than T_3 the merge factor has no effect. For regions smaller than T_3 the knowledge source virtually forces the candidate region to merge with one of the neighbors. The global effect of M_3 on the merge score is controlled by K_3.

Regions to be merged should exhibit "good" connectivity, hence it should be more difficult to merge regions with an extremely short common boundary. This fourth knowledge source serves to limit merges between regions with "poor" connectivity. The merge factor returned by this knowledge source is shown in figure 39d. The factor has no effect if the connectivity consists of more than T_4 edge segments. The merge factor is greater than 1.0 if the connectivity is less than T_4 edge segments. The global effect of M_4 on the merge score is controlled by K_4.

The knowledge sources used here represent a few very simple examples in a very large class of possible region-merge knowledge sources. Many additional knowledge sources based on different merge criteria are possible. Color differences of regions, differences in texture measures between regions, or even differences in current semantic interpretation probabilities could be used to control the merging process. It is interesting that the current simple operators were as effective in correcting oversegmentation as they were. Although it is necessary to set some thresholds, the algorithm is fairly stable as the thresholds are modestly shifted. Further development of this region merging paradigm may be quite worthwhile.

6.7 Experimental Results of Segmentation based on Clustering

This section shows the segmentation algorithm developed in this chapter applied to a number of different images. For each image we present the segmentations produced by selecting the clusters from a global histogram, by selecting the clusters from 32 by 32 subimages without benefit of the cluster addition algorithm, and by selecting clusters from 32 by 32 subimages using the cluster addition algorithm. For each of the three algorithms, the result of the algorithm with and without relaxation is also provided. In all cases the processing begins with the image smoothing pre-processing algorithm, followed by cluster selection, relaxation if applicable, removal of the artificial boundaries between subimages (none in the case of the global segmentation), and terminating with the region merging post-processing algorithm.

The region merging post-processing actually used in this section was an enhanced version of the merging algorithm described in section 6.6.2 which utilizes some additional knowledge sources. This version of the algorithm will be documented in a forthcoming technical report [Gri84].

These results are then followed by an experiment showing the importance of the image smoothing pre-processing for segmenting some images. This section ends with an analysis of some intermediate results which give a sense of the contributions and weaknesses of the various components in this algorithm.

All of the images used below (except the virus) were digitized from 35mm color slides to a spatial resolution of 512 by 512 with approximately 7 bits of dynamic range per color. The images were reduced to 256 by 256 pixel images by averaging and portions of the images were extracted for further processing. The first three images below (virus, bush, and window) are 64 by 64 pixel subimages while all of the other images are 128 by 128 pixel subimages of the corresponding 256 by 256 reduced image. In figure 41 we see the six segmentations of the image shown in figure 40. The left column of figure 41 contains segmentations produced without relaxation while the right column shows the segmentations produced after 40 iterations of the relaxation algorithm. The top row corresponds to the global clustering algorithm used by Nagin, the second row corresponds to the localized algorithm used by Nagin, and the bottom row corresponds to the cluster addition algorithm presented in this chapter. For the localized algorithms the artificial boundaries are conservatively removed as described in section 6.6.1 (using a local and global merge threshold of 1.2). Finally, for all algorithms the region merging algorithm is applied to produce the segmentations shown. We have not presented results from the cluster deletion knowledge sources since the region merging processing performs the needed merges more effectively since the merge algorithm makes the merge decision at the level of regions rather than at the level of clusters.

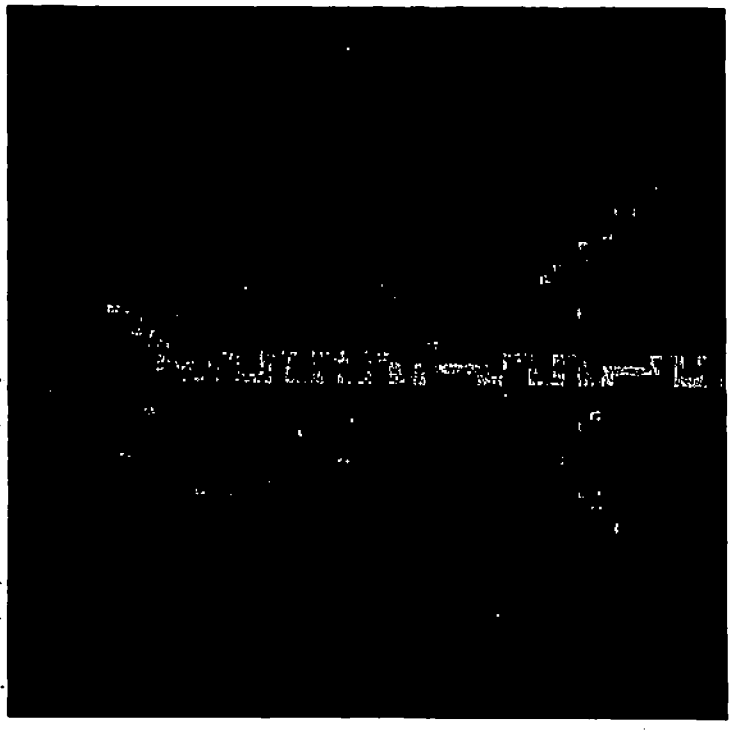


Figure 40: Intensity of Virus.

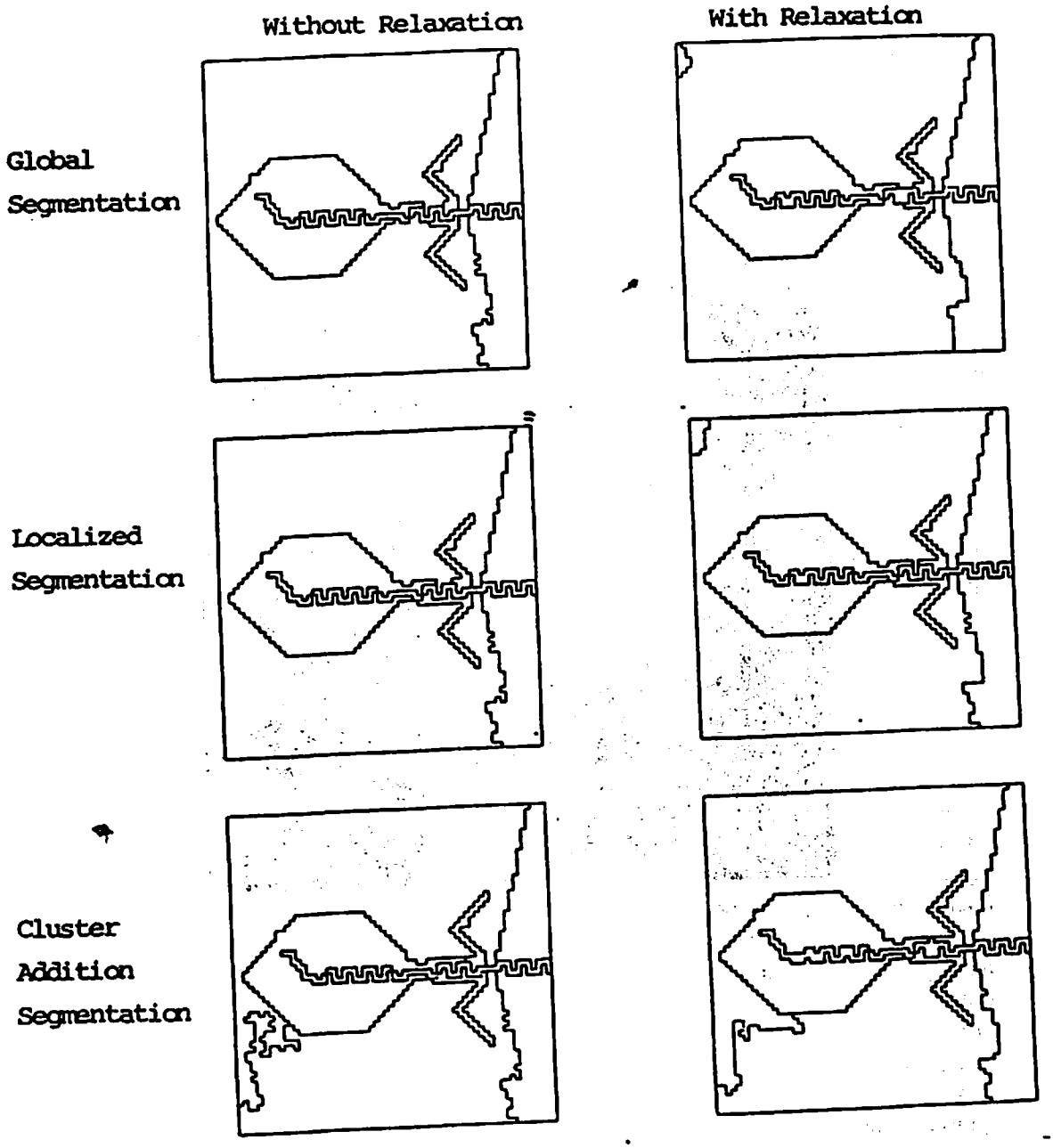


Figure 41: Segmentations of Virus.

Figures 43, 45, 47, 49, 51, 53, 55, and 57 show the corresponding segmentations applied to the images shown in figures 42, 44, 46, 48, 50, 52, 54, and 56. The segmentations produced by all these algorithms are surprisingly similar in most cases. Generally, the localized algorithm with cluster addition provides more detailed segmentations (without oversegmenting significantly) with fewer missing object boundaries than either the global or localized algorithms. There are cases where each of the other algorithms produces serious undersegmentations whereas the localized algorithm with cluster additions produced reasonable segmentations for all of the images.

The contribution of the very computationally expensive cluster relaxation algorithm is seen in the difference between the left and right columns of these figures. In the case of figure 57 the localized cluster addition result was computed in 355 cpu seconds without relaxation and in 4541 seconds for the segmentation with forty iterations of relaxation.[18] Given the subtle differences, it is difficult to justify this additional computational cost. Furthermore, although the relaxation may improve the quality slightly in some cases, it may, by shifting boundaries slightly, raise the variance of a region and thereby cause the artificial boundary removal algorithm or the postprocessing merge algorithm to incorrectly merge that region with one of its neighbors (this is what happens in the window area of figure 57), or the relaxation may blur region shape detail (as seen in figure 41).

In some cases the global cluster selection is better than localized cluster selection (see figures 51 and 55). This seems to be due to missing clusters in some subimages of the localized algorithm.

The effect of the image smoothing preprocessing is typically subtle, but figure 58 shows one case where the impact of the smoothing is dramatic. The six segmentations in this figure correspond to the segmentations of figure 41 but for which the image smoothing preprocessing was omitted. For this example, the smoothing made it possible to find the correct clusters to properly segment the image.

To provide additional insight into the relative contributions of the various processing stages, figures 59, 60, 61, 62, 63, and 64 show the intermediate segmentations produced for figure 56 using each of the results shown in figure 57. In other words we are presenting intermediate results of the six algorithms we have been comparing: global clustering, localized clustering, and localized clustering with cluster addition, each with and without relaxation. The upper left segmentation of each of these figures shows the segmentation after smoothing, clustering, and (if

[18] The run times are on a VAX-11/780 with floating point accelerator. The image is 128 by 128 pixels.

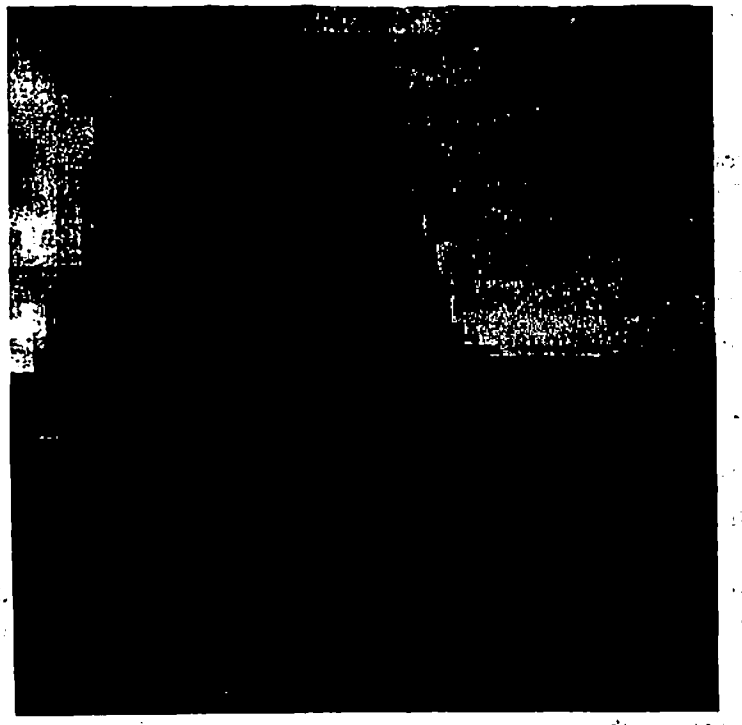


Figure 42: Intensity of Bush.

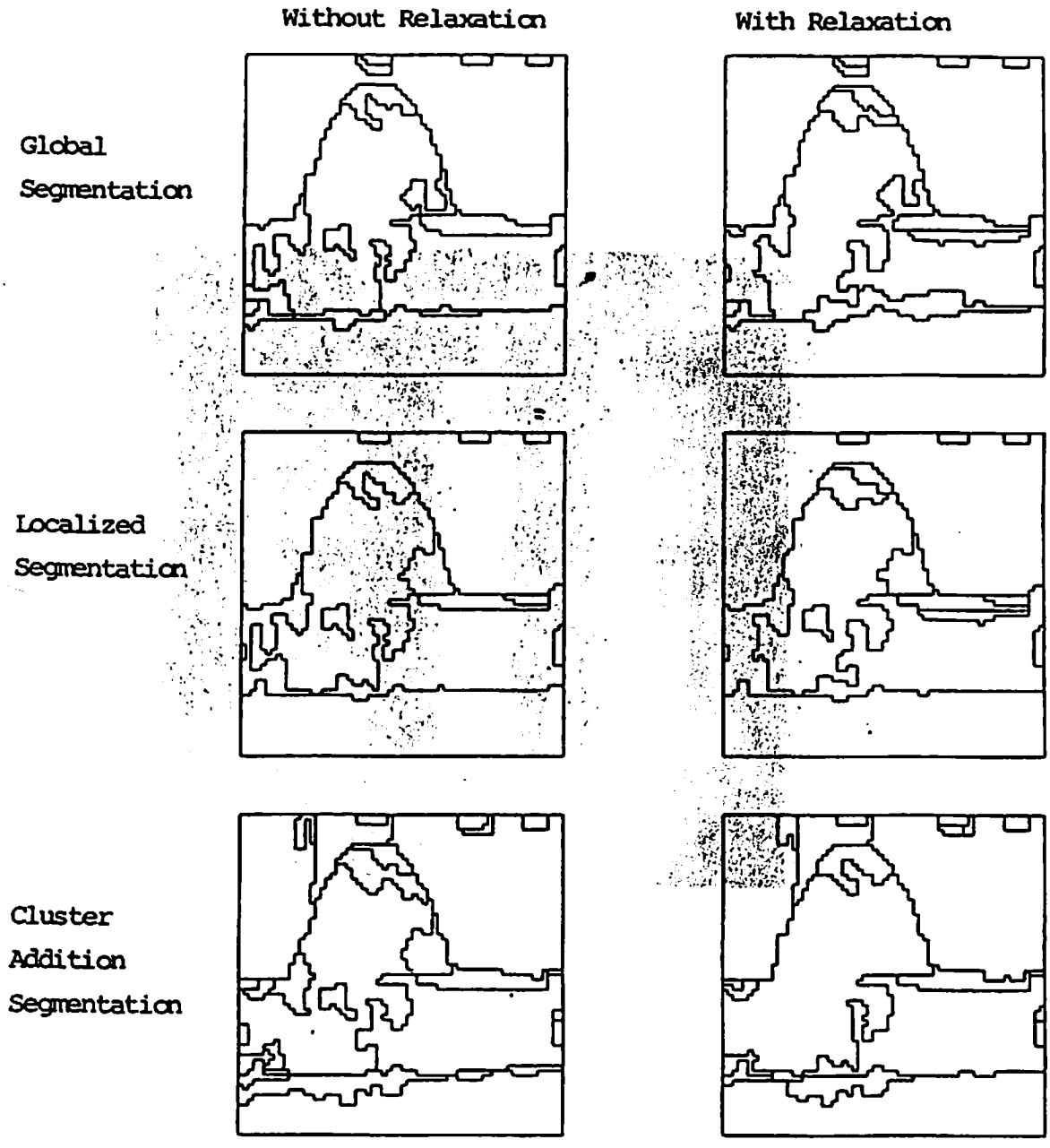


Figure 43: Segmentations of Bush.

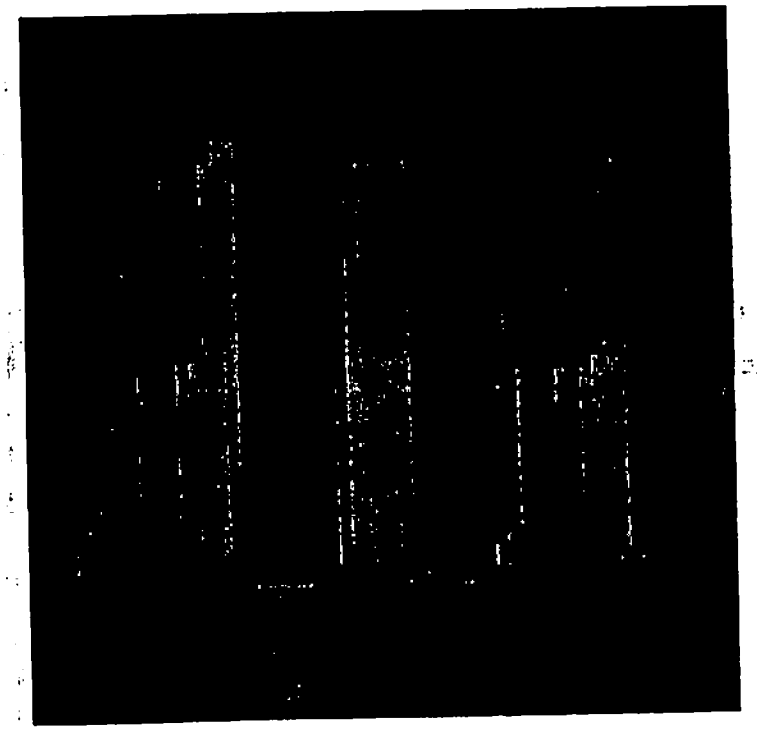


Figure 44: Intensity of Window.

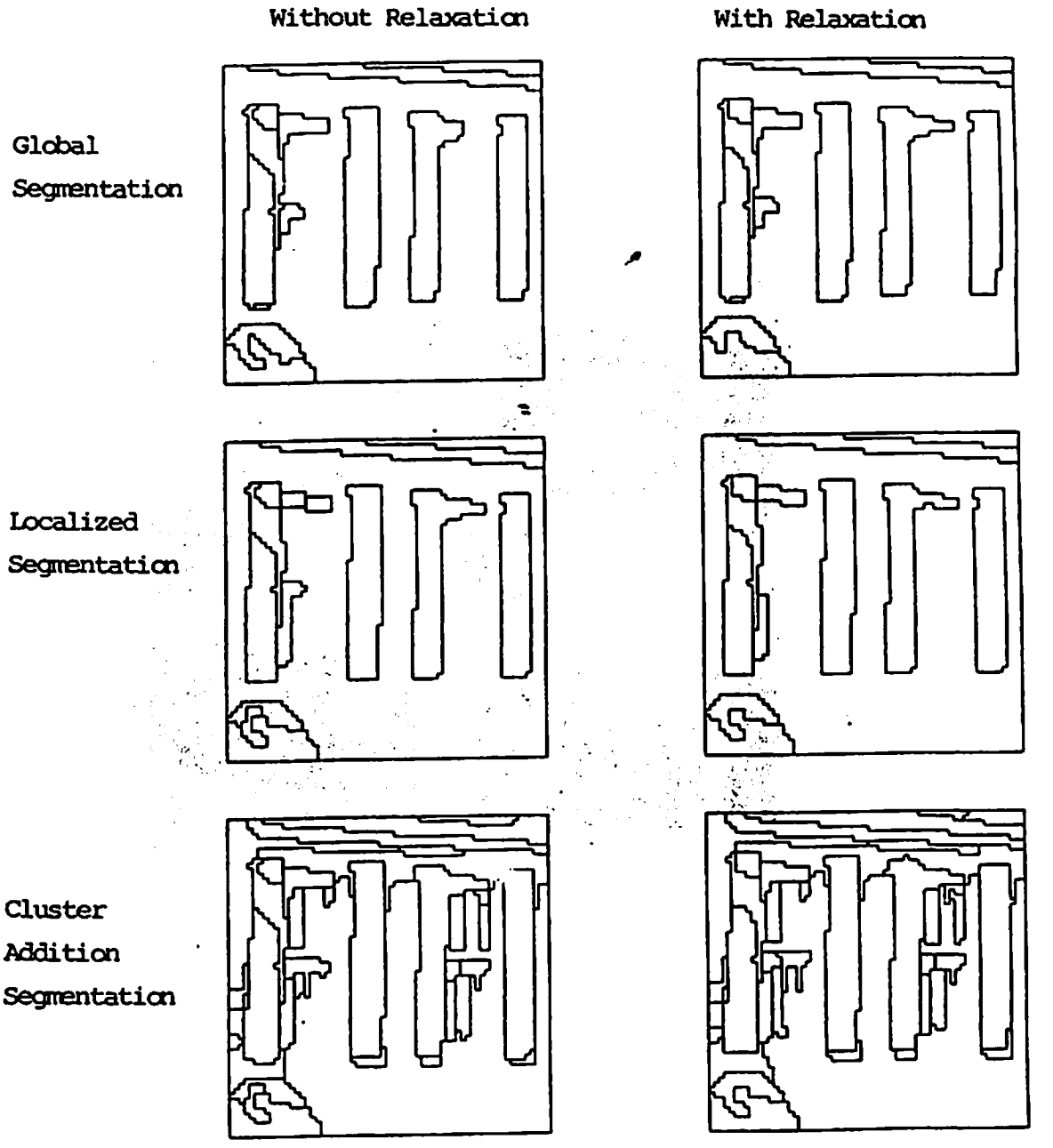


Figure 45: Segmentations of Window.

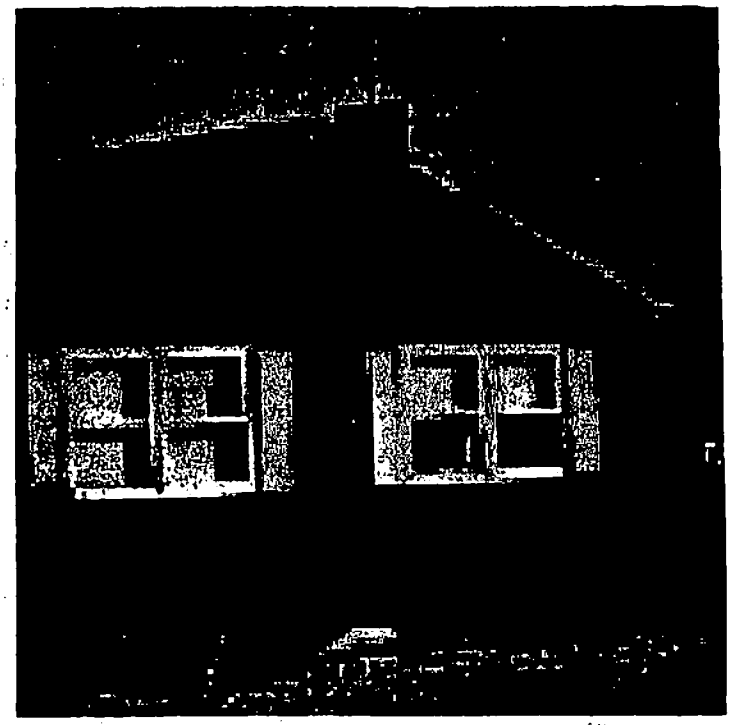


Figure 46: Intensity of House 1.

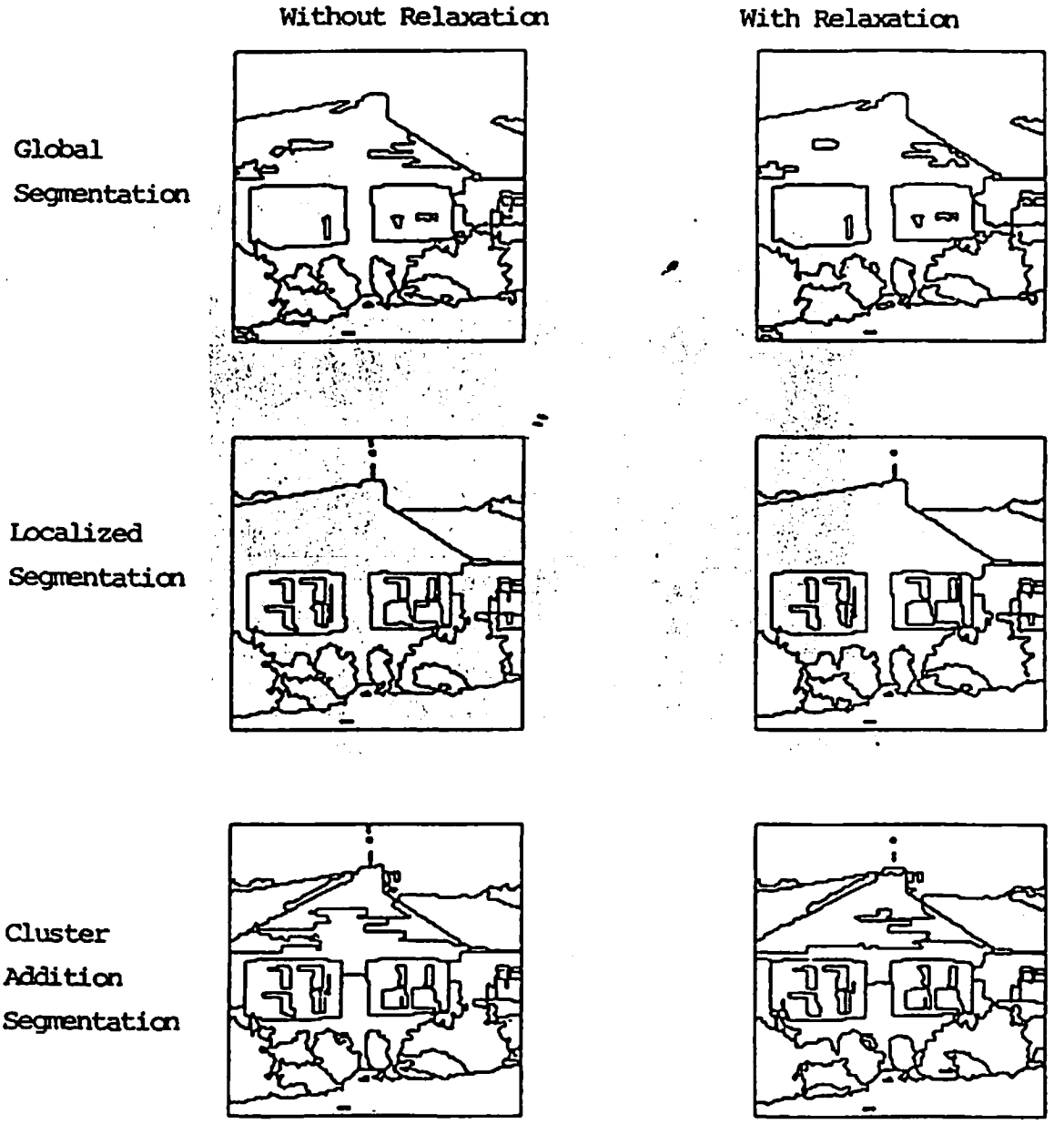


Figure 47: Segmentations of House 1.

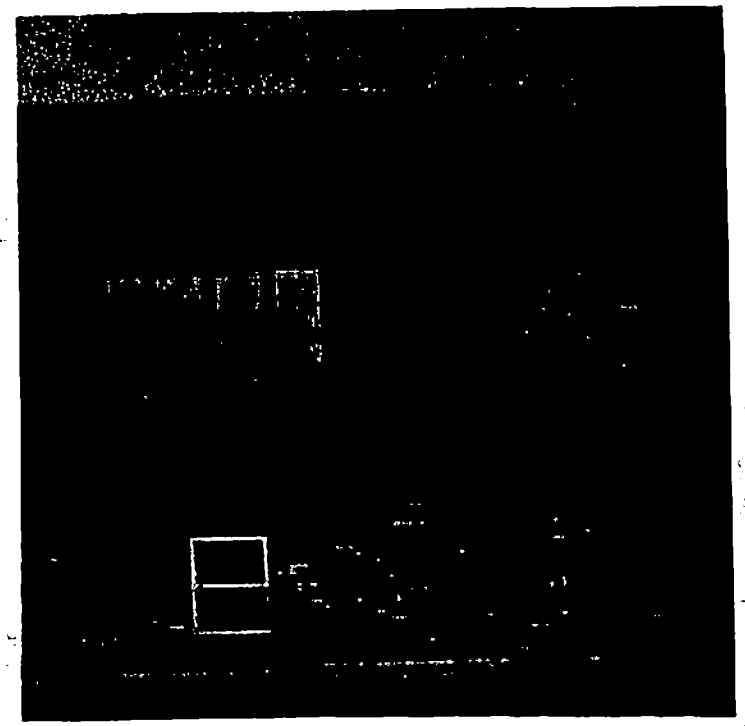


Figure 48: Intensity of House 2.

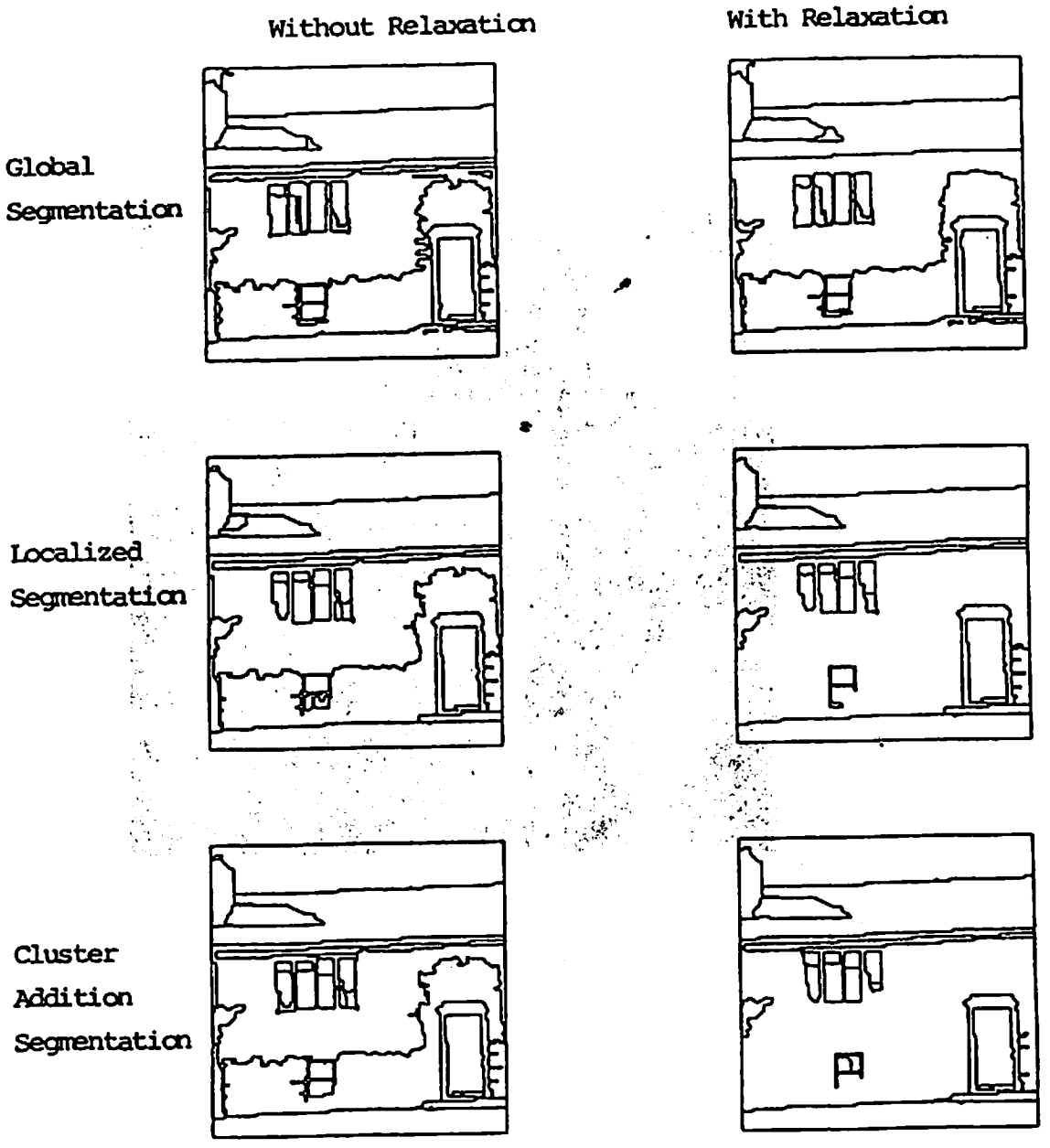


Figure 49: Segmentations of House 2.

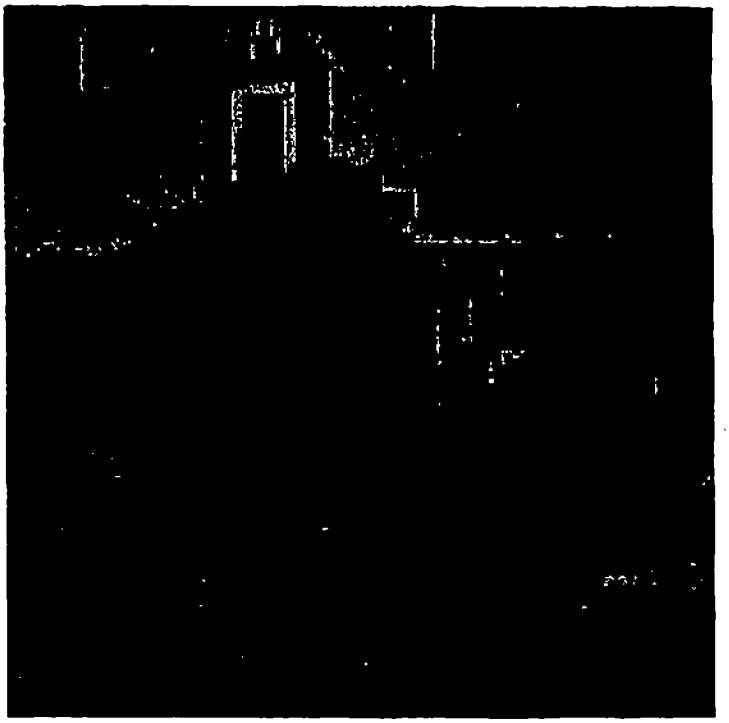


Figure 50: Intensity of House 3.

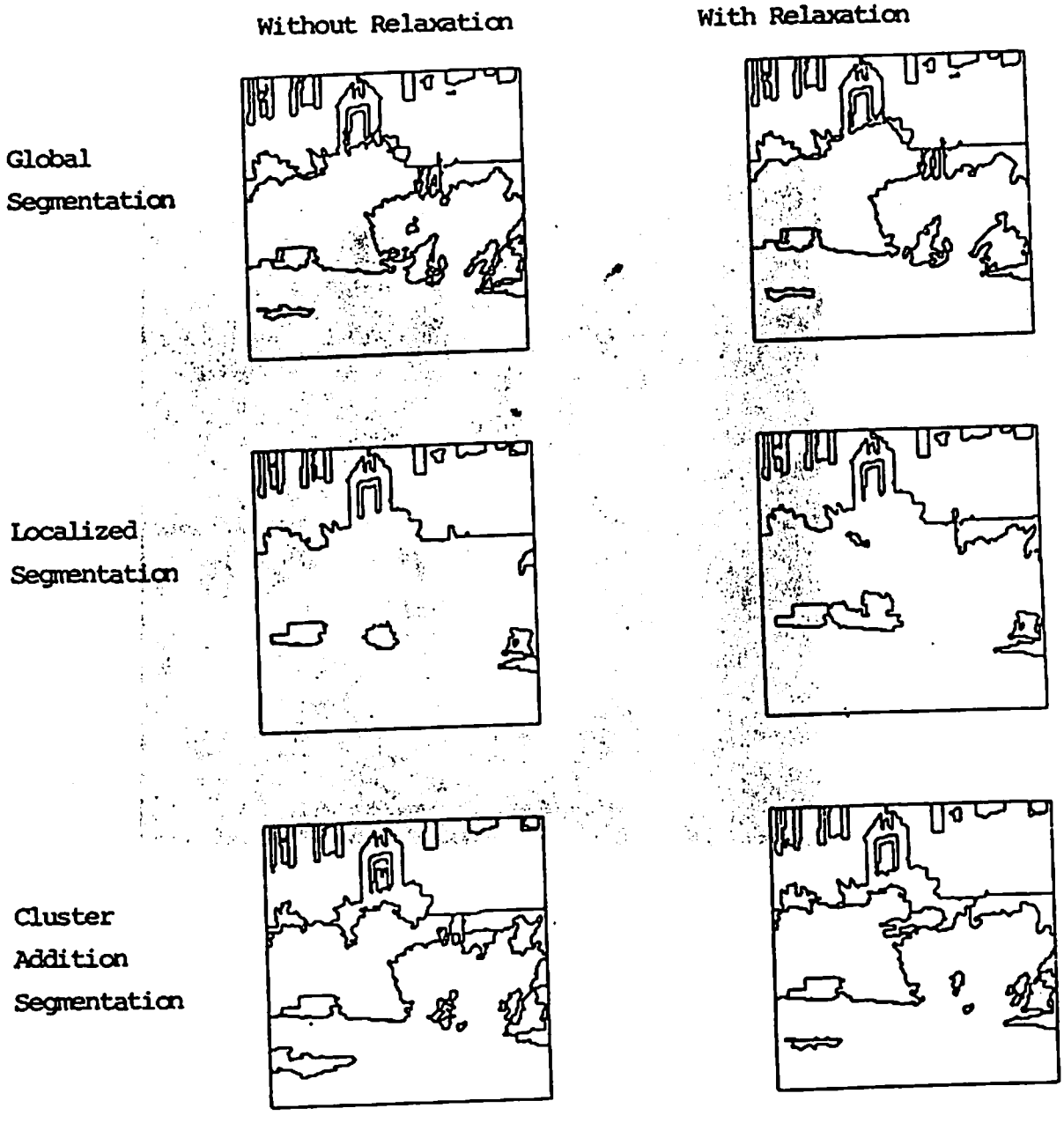


Figure 51: Segmentations of House 3.

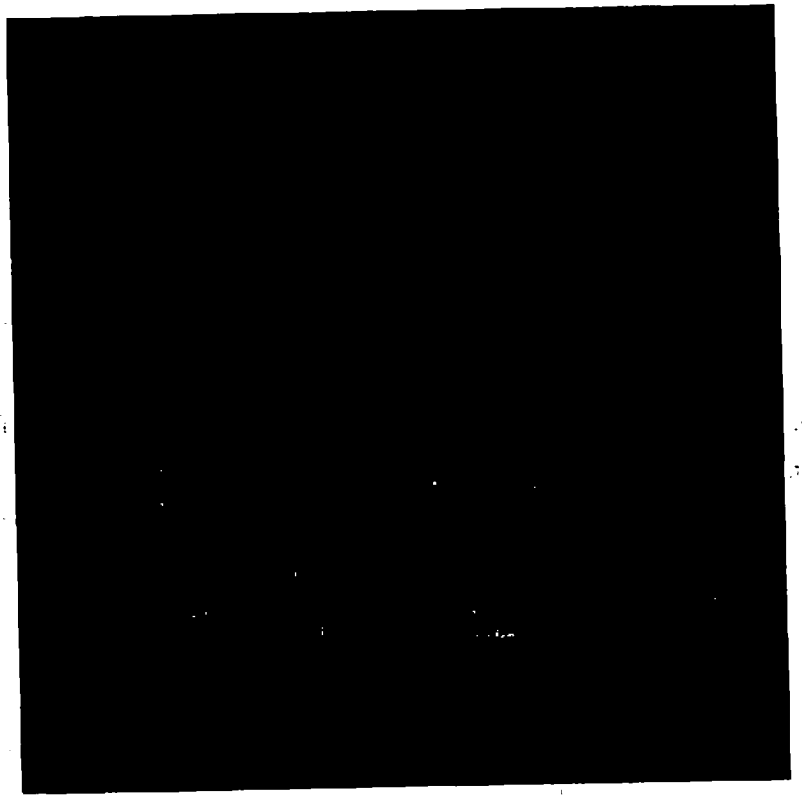


Figure 52: Intensity of House 4.

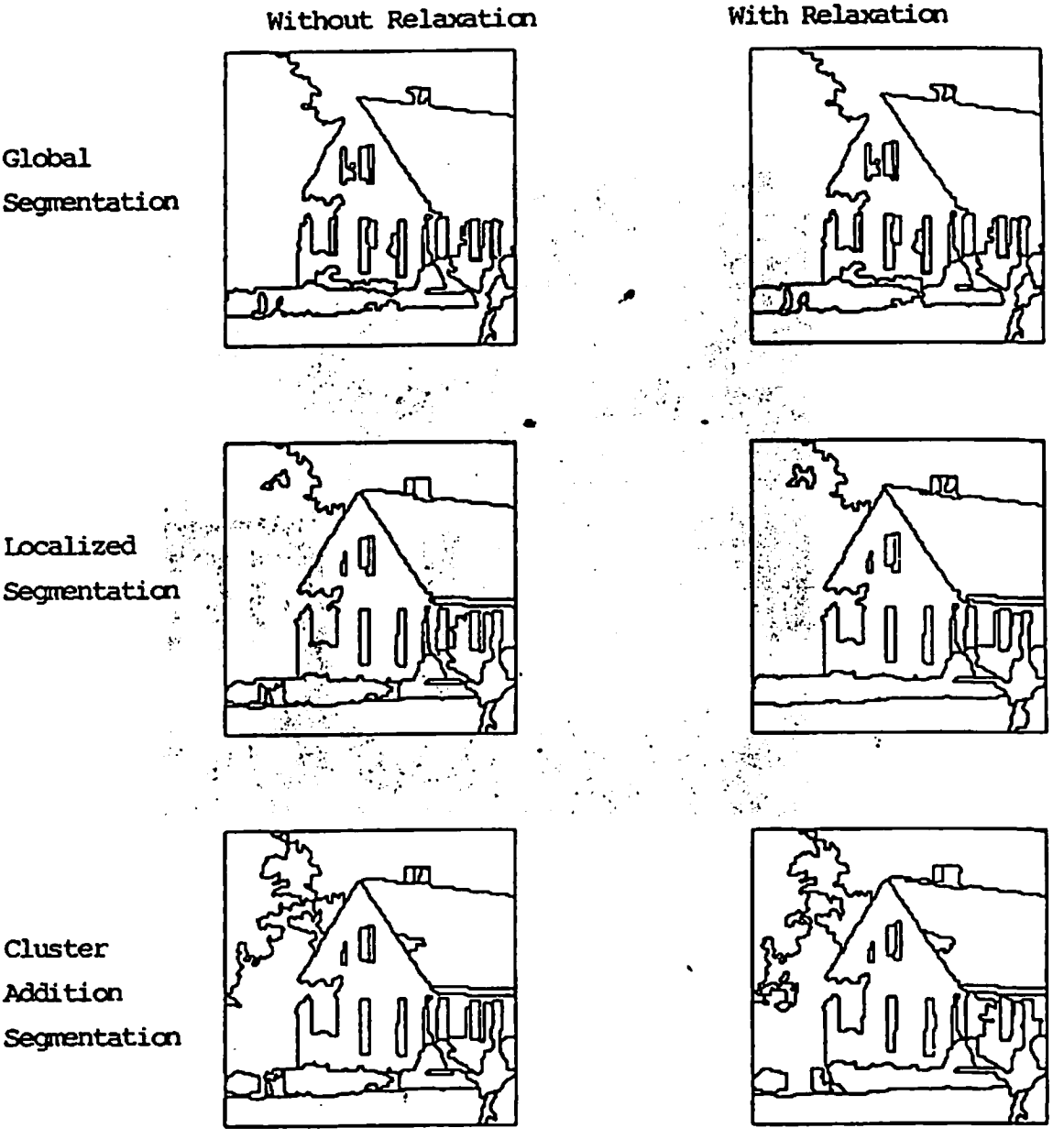


Figure 53: Segmentations of House 4.

Figure 54: Intensity of House 5.

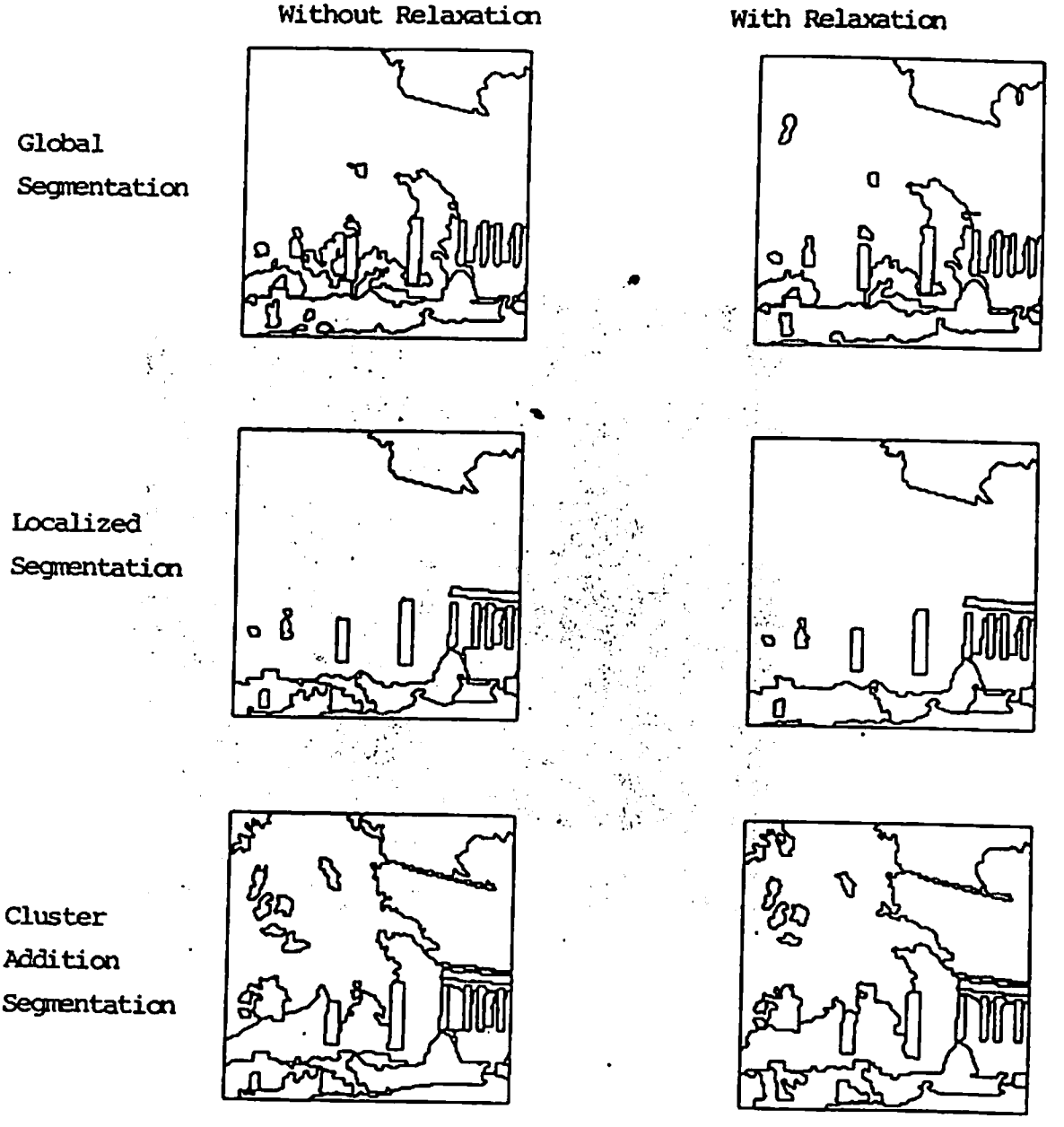


Figure 55: Segmentations of House 5.

Figure 56: Intensity of House 6.

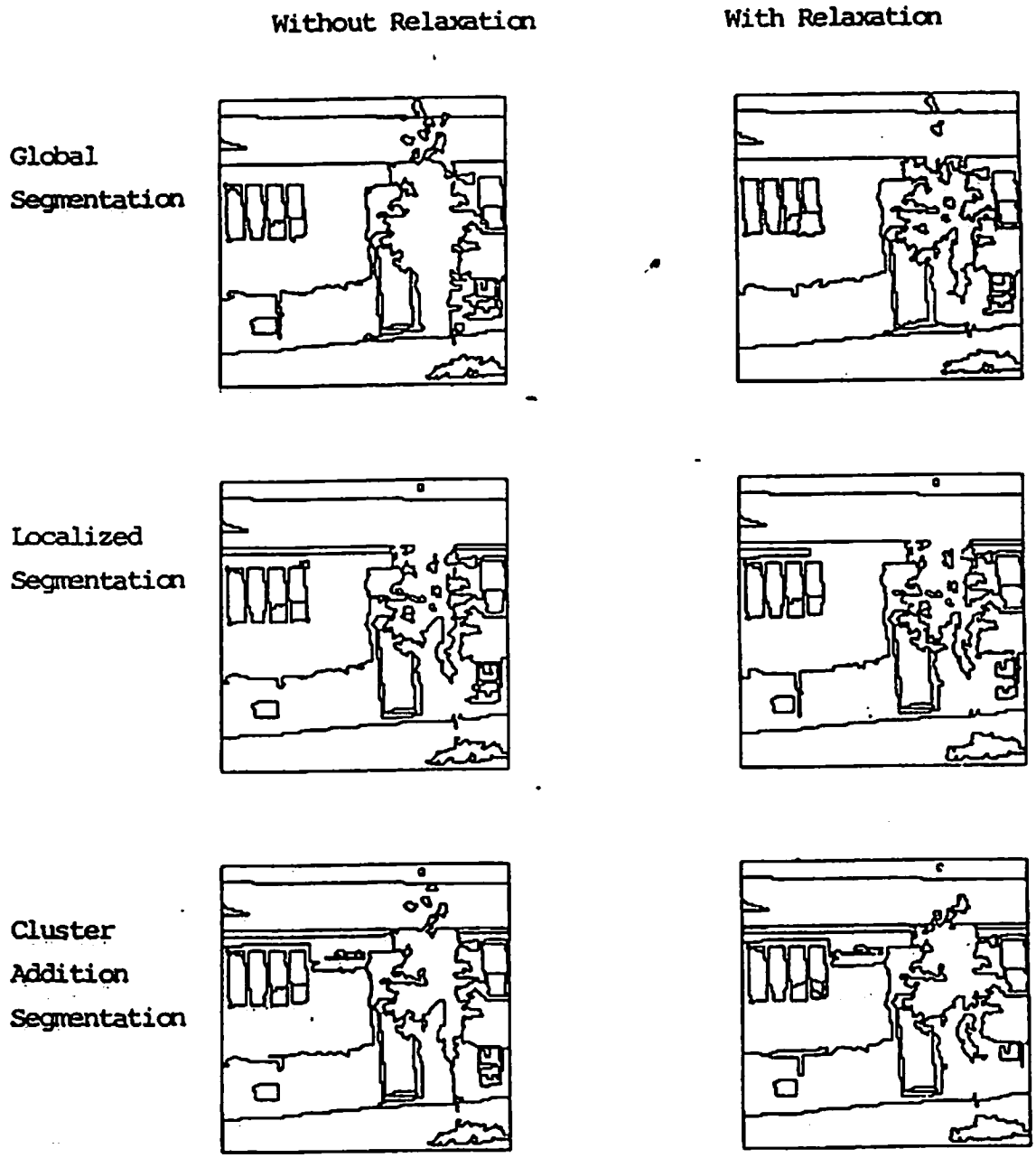


Figure 57: Segmentations of House 6.

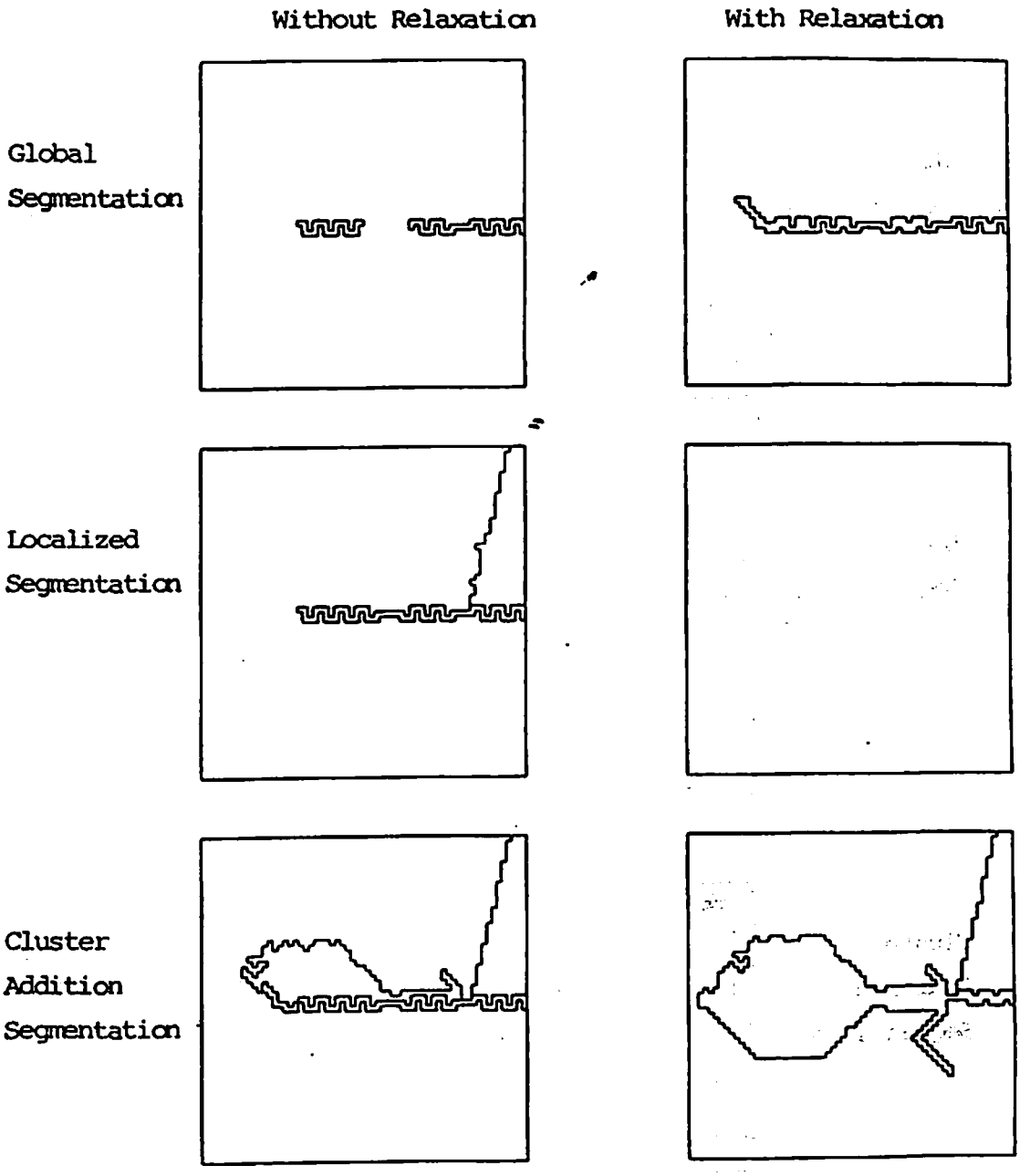
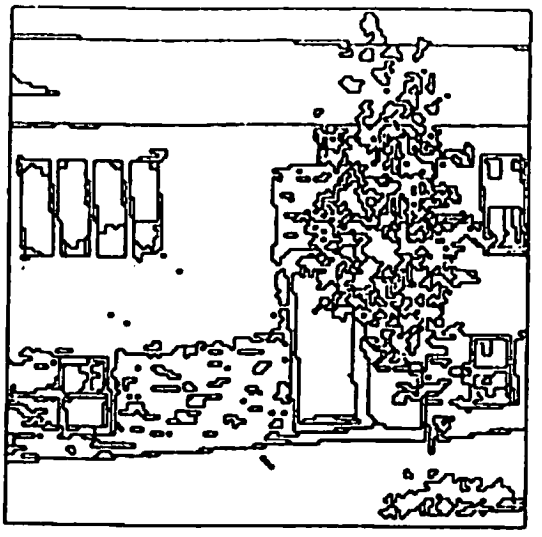
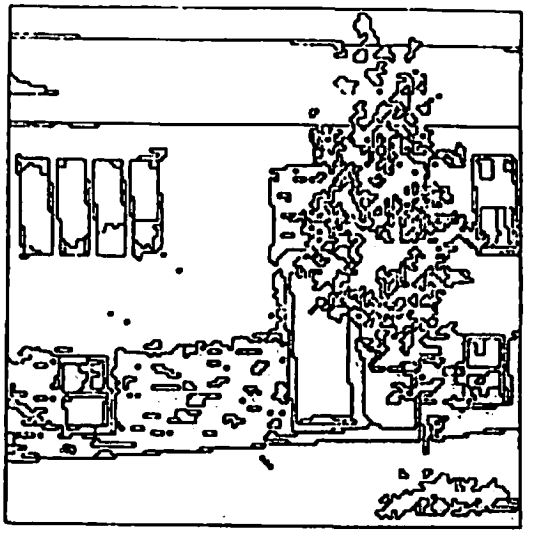


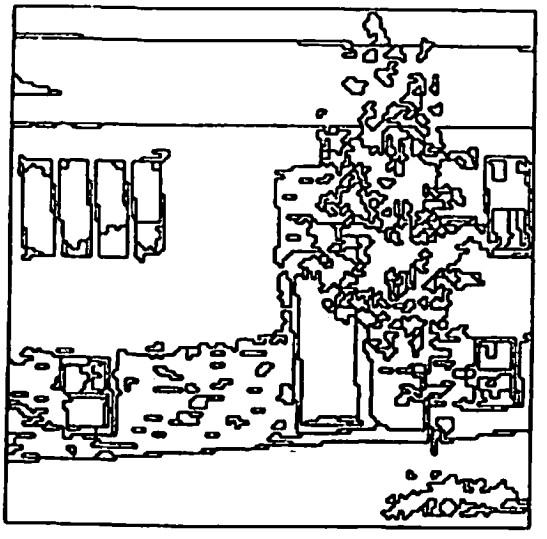
Figure 58: Segmentations of Unsmoothed Virus.



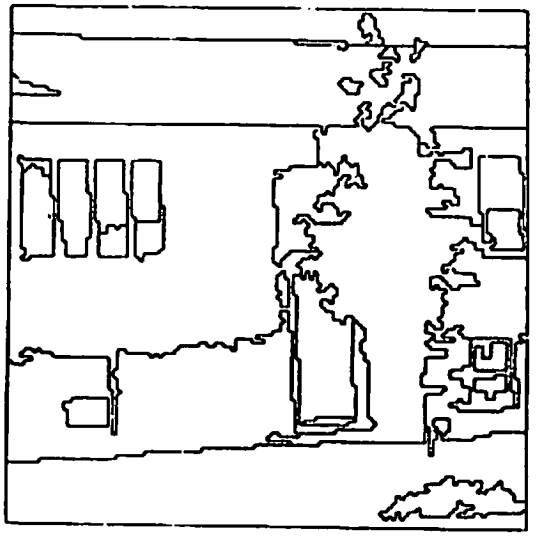
(a) Initial Segmentation



(b) Artificial Boundaries Removed

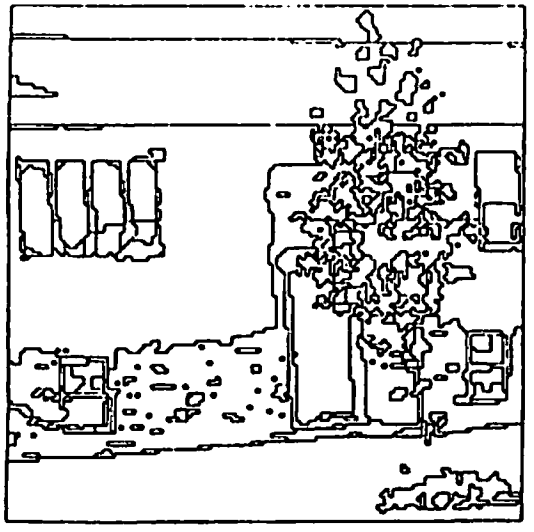


(c) One and Two Pixel Regions Removed

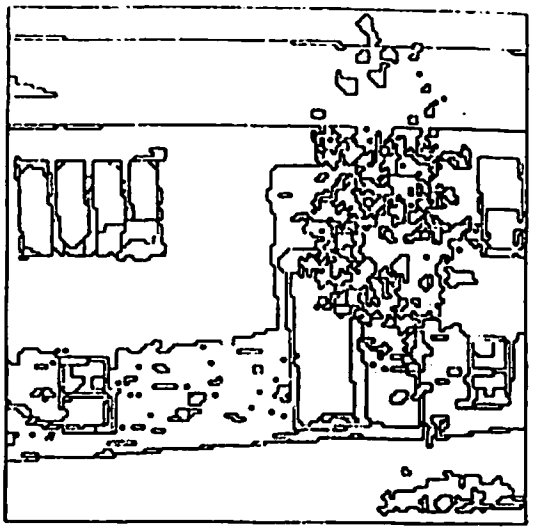


(d) General Region Merging

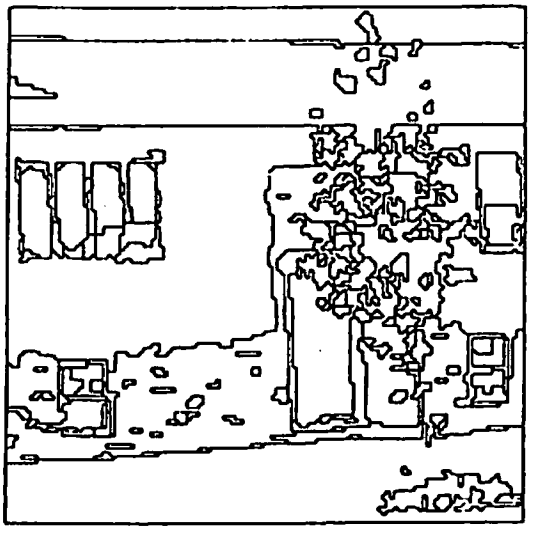
Figure 59: Segmentations of House 6 - Global Segmentation with No Relaxation.



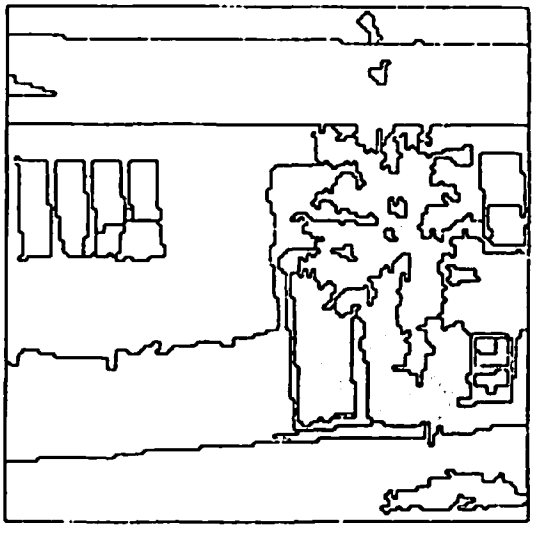
(a) Initial Segmentation



(b) Artificial Boundaries Removed

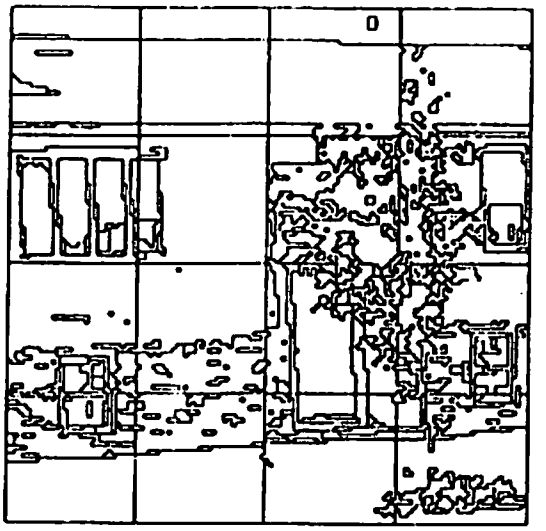


(c) One and Two Pixel Regions Removed

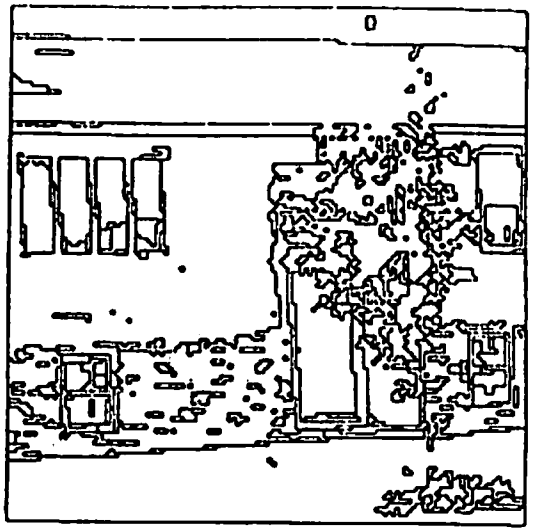


(d) General Region Merging

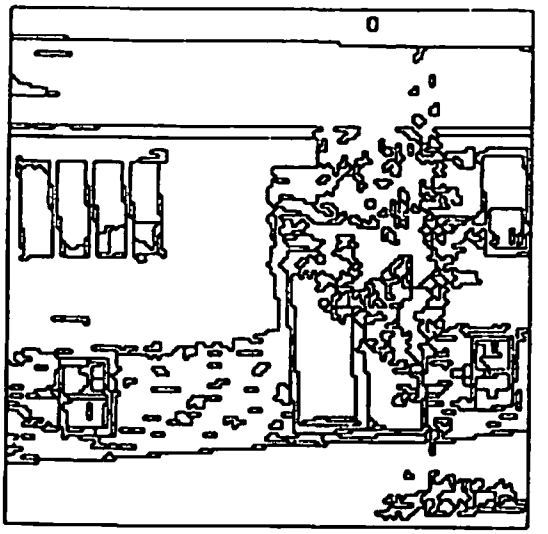
Figure 60: Segmentations of House 6 - Global Segmentation with Relaxation.



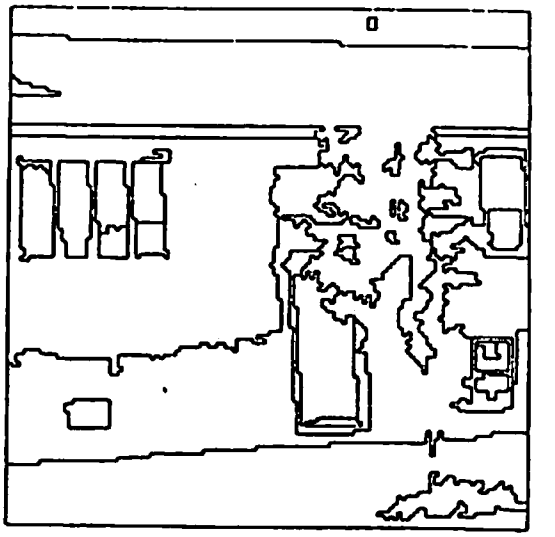
(a) Initial Segmentation



(b) Artificial Boundaries Removed

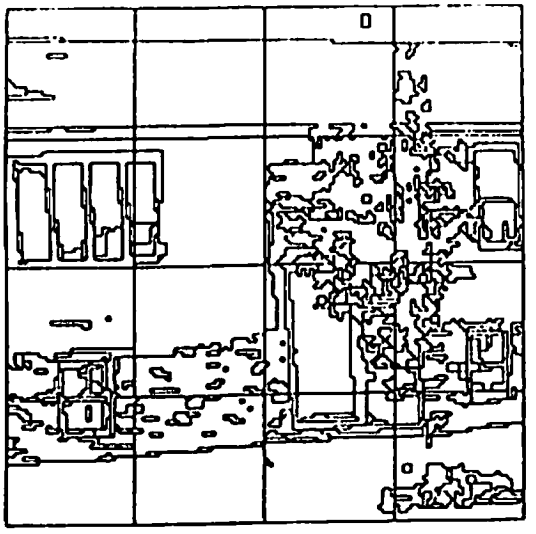


(c) One and Two Pixel Regions Removed

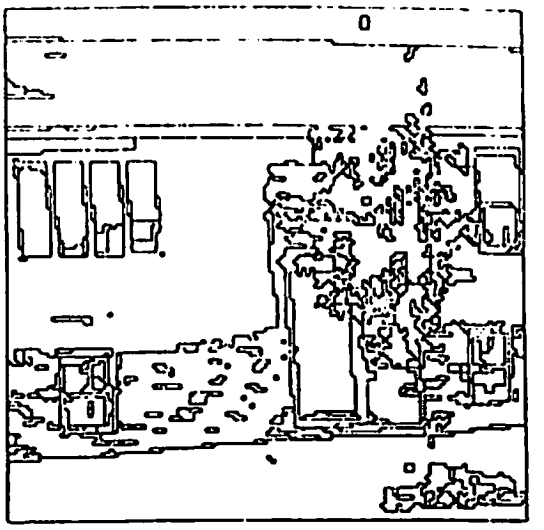


(d) General Region Merging

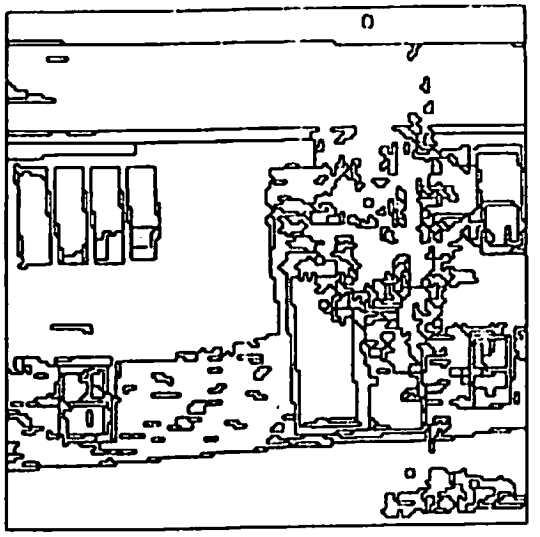
Figure 61: Segmentations of House 6 - Localized Segmentation with No Relaxation.



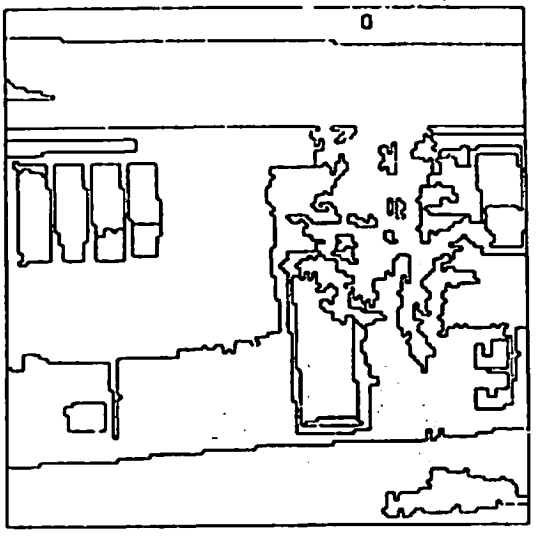
(a) Initial Segmentation



(b) Artificial Boundaries Removed

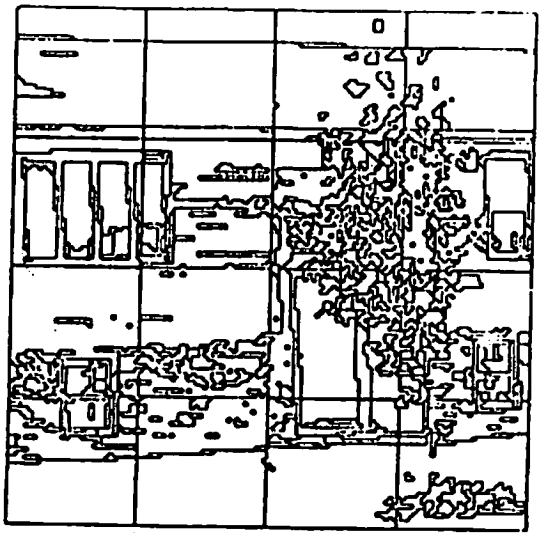


(c) One and Two Pixel Regions Removed

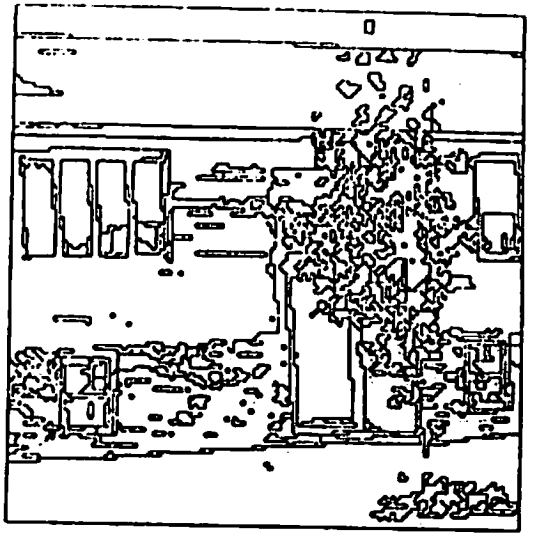


(d) General Region Merging

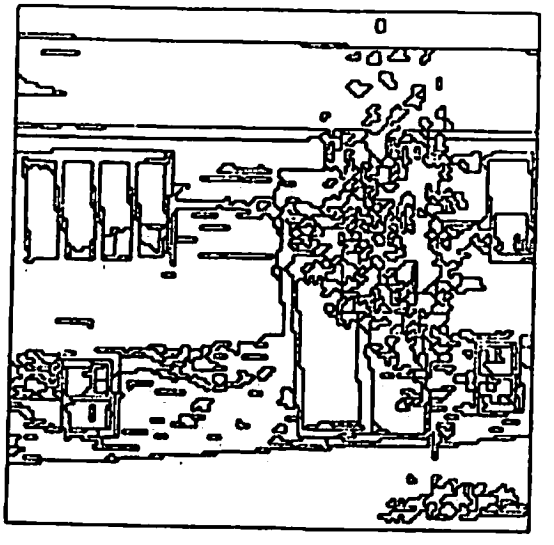
Figure 62: Segmentations of House 6 - Localized Segmentation with Relaxation.



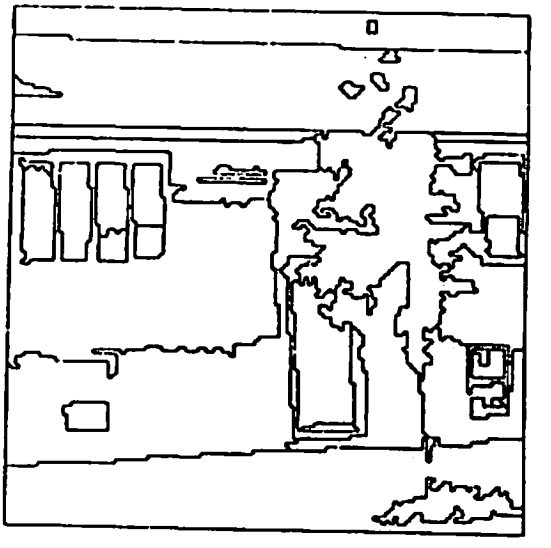
(a) Initial Segmentation



(b) Artificial Boundaries Removed

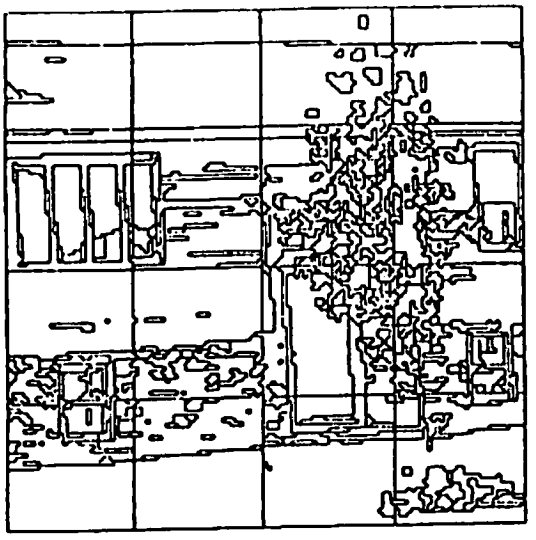


(c) One and Two Pixel Regions Removed

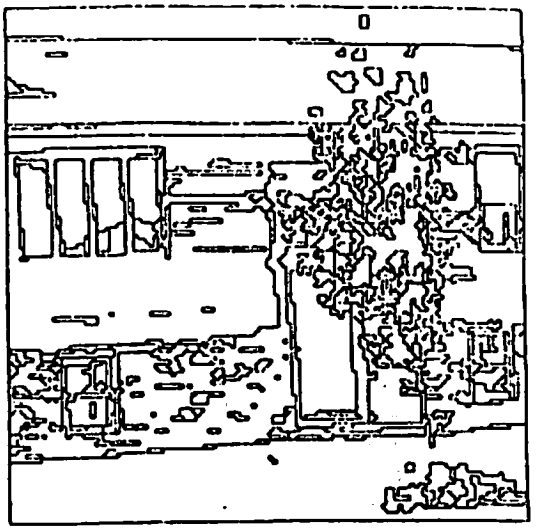


(d) General Region Merging

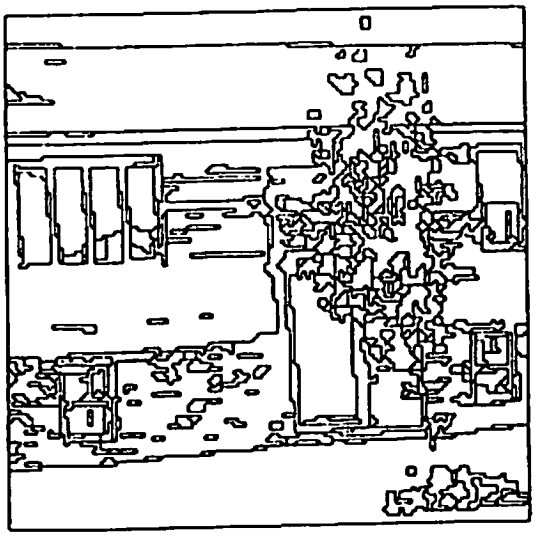
Figure 63: Segmentations of House 6 - Cluster Addition Segmentation with No Relaxation.



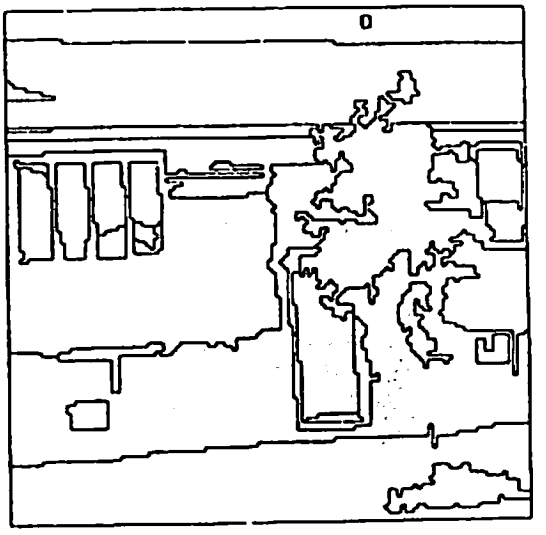
(a) Initial Segmentation



(b) Artificial Boundaries Removed



(c) One and Two Pixel Regions Removed



(d) General Region Merging

Figure 64: Segmentations of House 6 - Cluster Addition Segmentation with Relaxation.

applicable) relaxation, but before artificial boundaries due to the localization (if any) have been removed. The upper right image shows the segmentation after the artificial boundaries are removed (for figures 59 and 60 no such boundaries exist and the upper right image is identical to the upper left image). The lower left segmentation has had one and two pixel regions removed (by merging all regions of size 1 or size 2 with their locally most similar neighbor). The lower right image shows the final segmentation after the region merging process is applied. Figures 59 and 60 contain an example where the relaxation segmentation loses significant detail in the window area in the lower left corner of the image while the global segmentation without relaxation did not loose all of this detail. As can be seen, the information is not lost until the region merging algorithm is applied. Presumably, this is due to a slight shifting of the region boundaries through the relaxation which increases the internal variance of some regions allowing them to merge incorrectly with one of its neighbors. Figure 62 shows an example where a missing cluster in the top row and third column of subimages causes the merging of the roof and tree regions. The erroneous merge occurs in the algorithm which conservatively removed the artificial inter subimage boundaries introduced by the localization. In this example this algorithm was clearly not conservative enough.

6.8 Summary of Segmentation based on Clustering

This chapter has addressed issues of integrating non-semantic knowledge into a segmentation algorithm. We have defined the concept of a *segmentation process instantiation* to include image dependent parameter selection and feature selection for a particular segmentation algorithm. We applied this concept, utilizing different types of non-semantic knowledge, to improve upon the cluster selection component of the modified Nagin segmentation algorithm.

By examining particular types of image segmentation problems, we demonstrated the value of combining multiple types of non-semantic knowledge to improve the segmentation process. In particular, we have addressed issues of undersegmentation due to missed clusters and oversegmentation due to clusters derived from micro-texture, intensity gradients, and "mixed pixel" regions in the modified histogram cluster based segmentation algorithm. Addressing these issues requires us to deal with the relationship between feature space analysis and local and global characteristics of image space.

This chapter also includes a region merging algorithm which makes region merge decisions based on a number of merge factors, each of which is based on somewhat different non-semantic knowledge.

The result of integrating the components discussed in this chapter led to a much more robust segmentation algorithm which generally produces somewhat more detailed segmentations than the algorithms to which it is compared. The most important observation in section 6.7 was that both of the comparison algorithms

produced very poor segmentations on some of the images, while the integrated algorithm produced reasonable segmentations for every image tested.

7.0 COMBINING INDEPENDENT SEGMENTATION PROCESSES

In chapter 6 we explored ways in which multiple sources of knowledge could be integrated into a single segmentation algorithm. In this chapter we will consider how several separate algorithms, each based on different knowledge or assumptions about the world, might be integrated to produce a single unified segmentation.

Given a set of segmentation processes, each of which computes a segmentation of the same scene using different features and algorithms, one would expect these segmentations to be somewhat different but to share many of the same boundaries. In section 7.1 we first consider ways in which the segmentation results might be combined without directly unifying the algorithms, while in section 7.2 we consider the possibility of dynamically integrating the algorithms such that mutual feedback leads to unified algorithms with a single, hopefully consistent, segmentation.

7.1 Static Integration of Segmentations

This section considers two methods by which a set of segmentation results of the same image might be combined into a single segmentation. These methods operate without any prior knowledge of the characteristics of the algorithms which generated the segmentations.

The first method is based on the region merging algorithm presented in section 6.6.2. Here, all region boundary and edge hypotheses proposed by any of the segmentation processes are combined and the merging process is used to remove some of these boundaries. The second method is based on the concept of "islands of reliability" from Hearsay. Here, boundaries of the image which obtain support from many of the contributing segmentations are considered to be correct and these boundaries are iteratively extended using boundary continuity criteria. It is argued that this method could easily be extended to allow integration of semantic expectations generated by high level interpretation processes or prior knowledge about the behavior of specific segmentation algorithms in image areas with known characteristics into the unification process.

Sections 7.1.1 and 7.1.2 below examine some segmentations produced by three distinct segmentation algorithms applied to several image features. The intent is to provide the reader with insights into the differences between the characteristics of the segmentation algorithms and the differences among segmentations based on

different features. If all of the segmentations, regardless of image feature or segmentation algorithm, produced exactly the same segmentation errors, then no integration of these segmentations could hope to correct the errors. Fortunately, this does not seem to be the case.

In the first set of examples in the following section, the region relaxation algorithm, the edge relaxation algorithm, and the multi-thresholding algorithm were applied to three intensity images. The examples of section 7.1.1 will show that the algorithms do produce different segmentations and the errors made by the algorithms are not necessarily identical. In sections 7.1.3 and 7.1.4 we will attempt to avoid some of those errors, but for now, let us summarize with some observed or deduced characteristics of the three algorithms.

The region cluster labeling algorithm seems prone to oversegmentation, especially in areas of texture such as trees. The algorithm forms regions and therefore, there are no problems with unclosed contours as in the edge algorithm. Since the algorithm is dependent on feature space clusters, the segmentation may miss boundaries even when there is local high contrast evidence for the boundary, while boundaries may be introduced in areas where no edge is discernable.

The multi-threshold algorithm is sensitive to strong boundaries and rarely misses them, but the algorithm may also introduce boundaries even if there is little or no local evidence to support such a boundary. Furthermore, this algorithm is sensitive to high contrast micro-texture and may find multiple representations for blurred boundaries. This algorithm also leads to closed regions.

The edge relaxation algorithm does not form closed contours and may fail to detect low contrast boundaries. The algorithm tends to oversegment in the presence of texture. The algorithm finds boundaries only where there is local contrast support (unlike the other two algorithms which may introduce boundaries to form closed regions even if no evidence for the boundary exists).

Multiple Segmentation Examples.

Figures 65, 66, and 67 show three intensity images and their corresponding segmentations. Figure 65a shows the virus image with a gradient across the background. Figure 65b shows the segmentation resulting from the region relaxation algorithm. The resulting segmentation has few errors. Figure 65c shows the result of applying the edge relaxation algorithm. In this segmentation, a number of micro-texture regions are segmented as separate regions, and many significant boundaries fail to form closed regions. Furthermore, some low contrast but significant boundaries are lost entirely. Figure 65d shows the segmentation produced by the multi-threshold algorithm. This segmentation exhibits some gross errors in the areas where the local contrast of the desirable boundaries is small due to the intensity gradient.

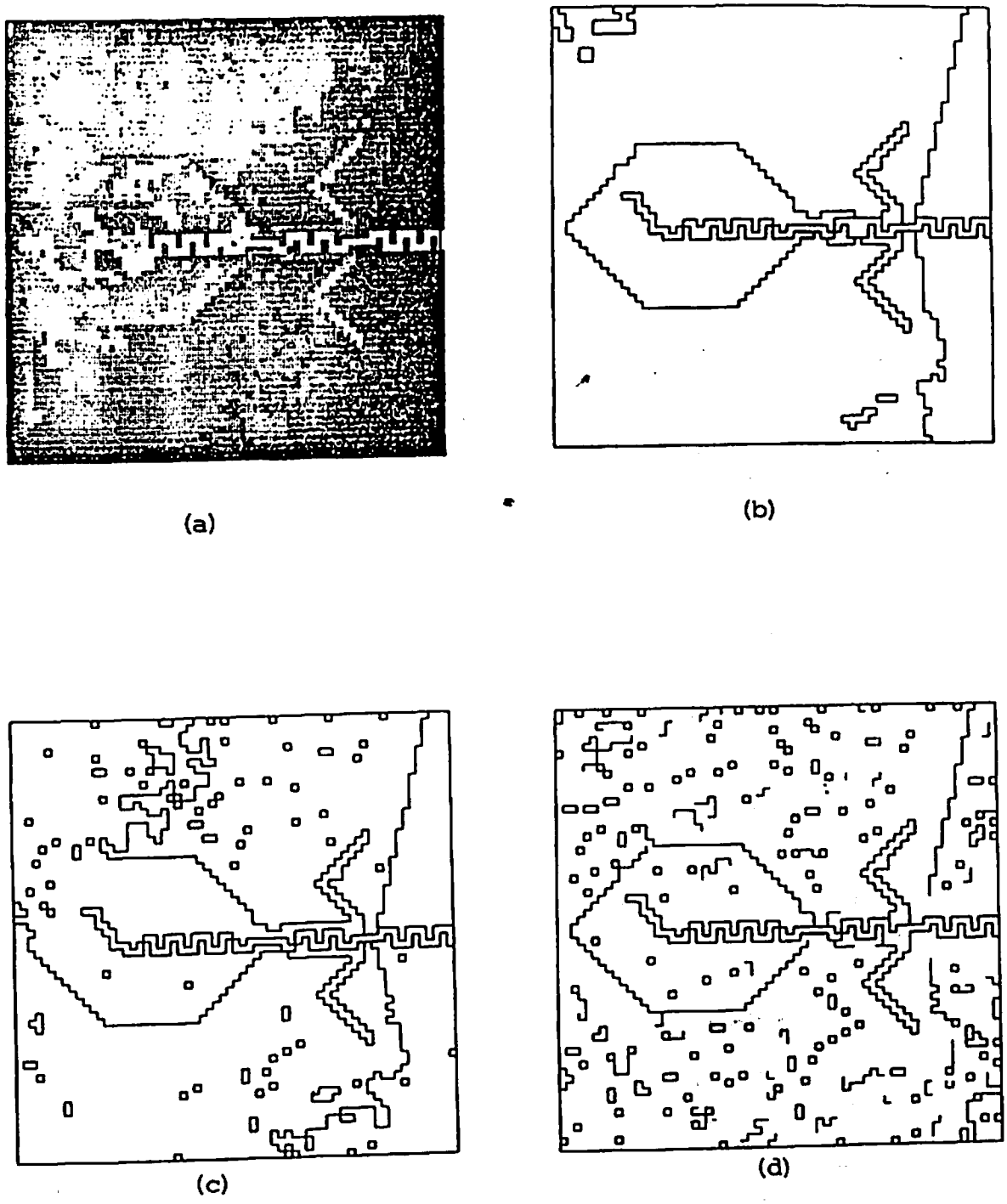


Figure 65: Virus Segmentations for Cluster, Edge, and Threshold Algorithms.

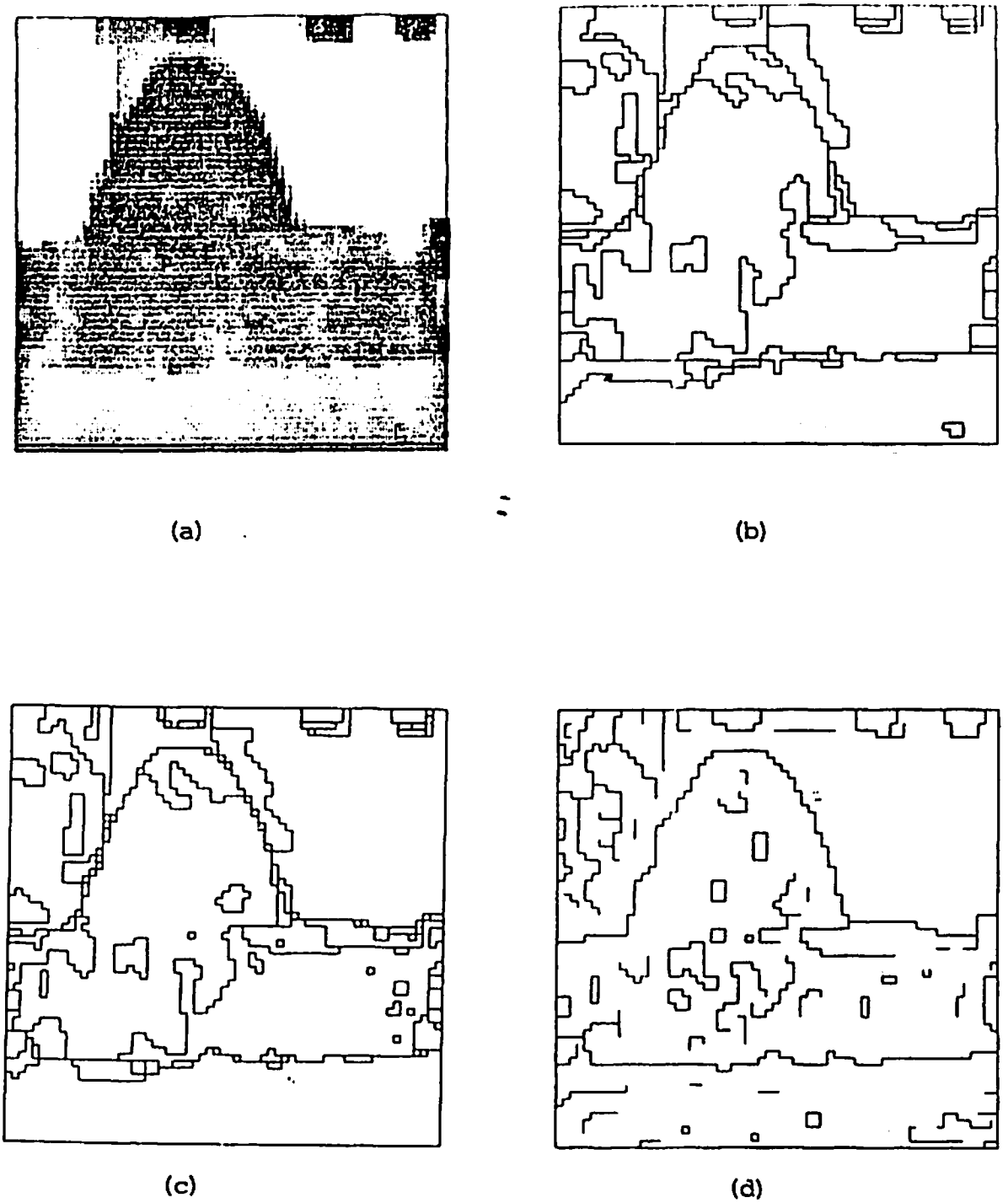


Figure 66: Bush Segmentations for Cluster, Edge, and Threshold Algorithms.

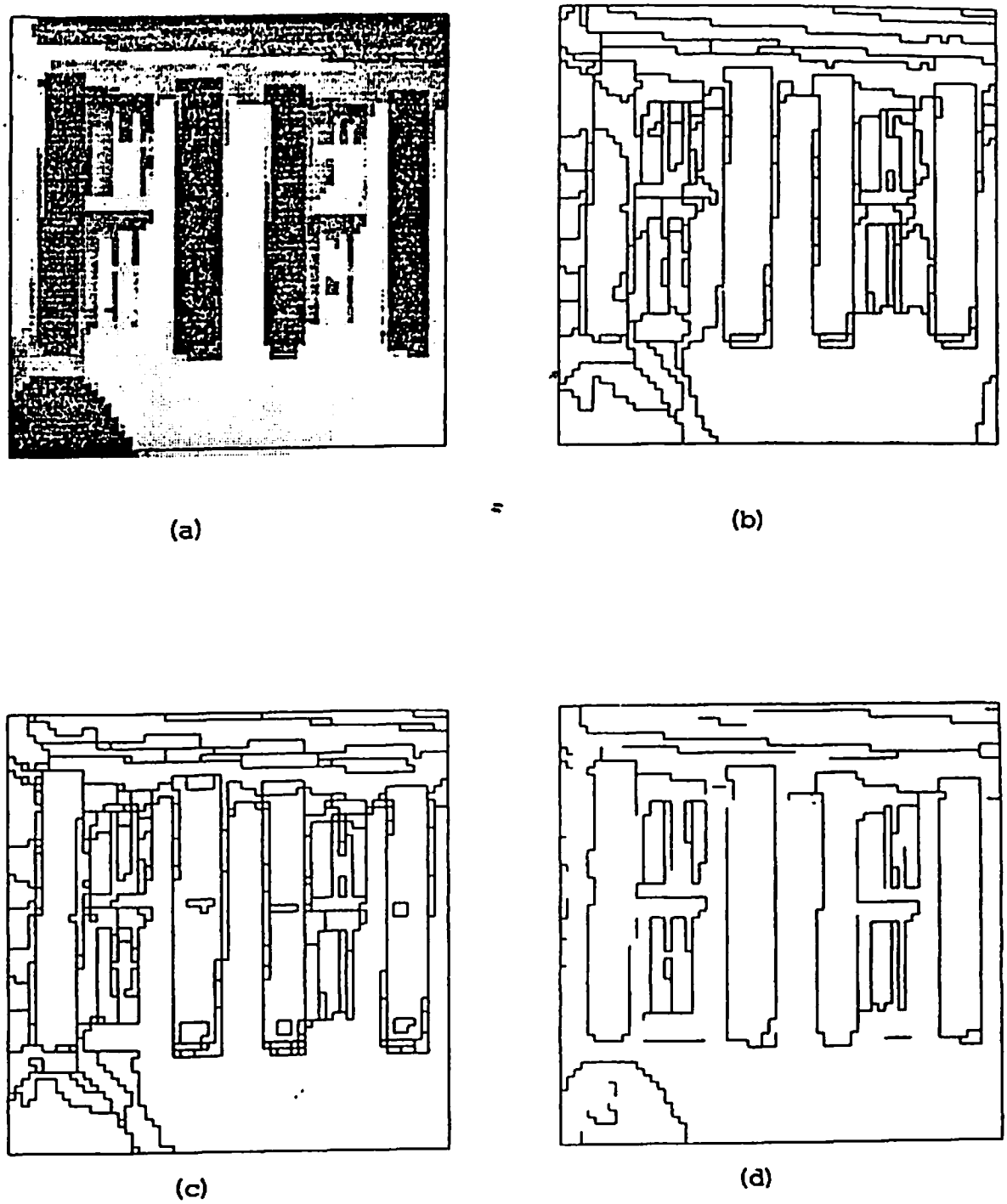


Figure 67: Window Segmentations for Cluster, Edge, and Threshold Algorithms.

In figure 66b we see a certain amount of oversegmentation due to intensity gradients around the window bottoms and the top of the bush, as well as the tree shadow on the left house wall. Figure 66c again shows some boundaries corresponding to micro texture and many unclosed regions. Figure 66d shows multiple thresholds detecting the somewhat fuzzy boundary between the bush and house. Figure 67b and 67d seem to be quite similar in character and again are an oversegmentation which includes some mixed pixel regions at the shutter boundaries and considerable partitioning of the window interior. In the case of figure 67c we detect most of the window boundaries, but we still do not form closed boundaries.

A combined segmentation can be obtained from the individual segmentations in a variety of ways. A conservative approach would be to require all algorithms to agree on a boundary before accepting a boundary. Similarly, this requirement could be relaxed requiring m of n segmentations to agree on a boundary before accepting the boundary in the combined segmentation. To demonstrate the kind of results one could expect from such simple integration algorithms, we overlay the segmentations produced by the three algorithms in the previous examples, as shown in figure 68. The left column shows the intersection of the three segmentations for each of the three images. As can be seen, this intersection contains very few boundary segments which do not correspond to meaningful boundaries. The second column of figure 68 shows the boundaries agreed to by at least two of the three algorithms, while the last column shows all boundaries suggested by any of the three algorithms (the logical "or" or union of the segmentations). As can be seen the agreement of two or more segmentations leads to quite a few boundaries which may not be desirable in the final segmentation, while the "or" of the segmentations results in a gross oversegmentation.

Color Segmentation Examples.

Color information can contribute significantly to segmentation. Nagin advocated the use of a two-dimensional histogram of two color features (such as pairs of opponent color transforms) as the basis for the cluster labeling region algorithm. In this section we will show several segmentations of some of the images of the last section computed independently on each of the three raw color bands (red, green, and blue).[19] In the natural scene examples, the intensity at each pixel correlates extremely well with the raw color components. In these examples, the vast majority of the information critical to computing a "good" segmentation is available in the intensity image (as evidenced by the segmentations of the previous section). For this reason, we have also included an artificial image (generated by an image synthesis program using a complex lighting model) in which color information is critical in obtaining a reasonable segmentation.

---

[19] Our preliminary experiments with alternative color spaces were disappointing, and therefore, for simplicity, we limit this discussion to the raw color components.

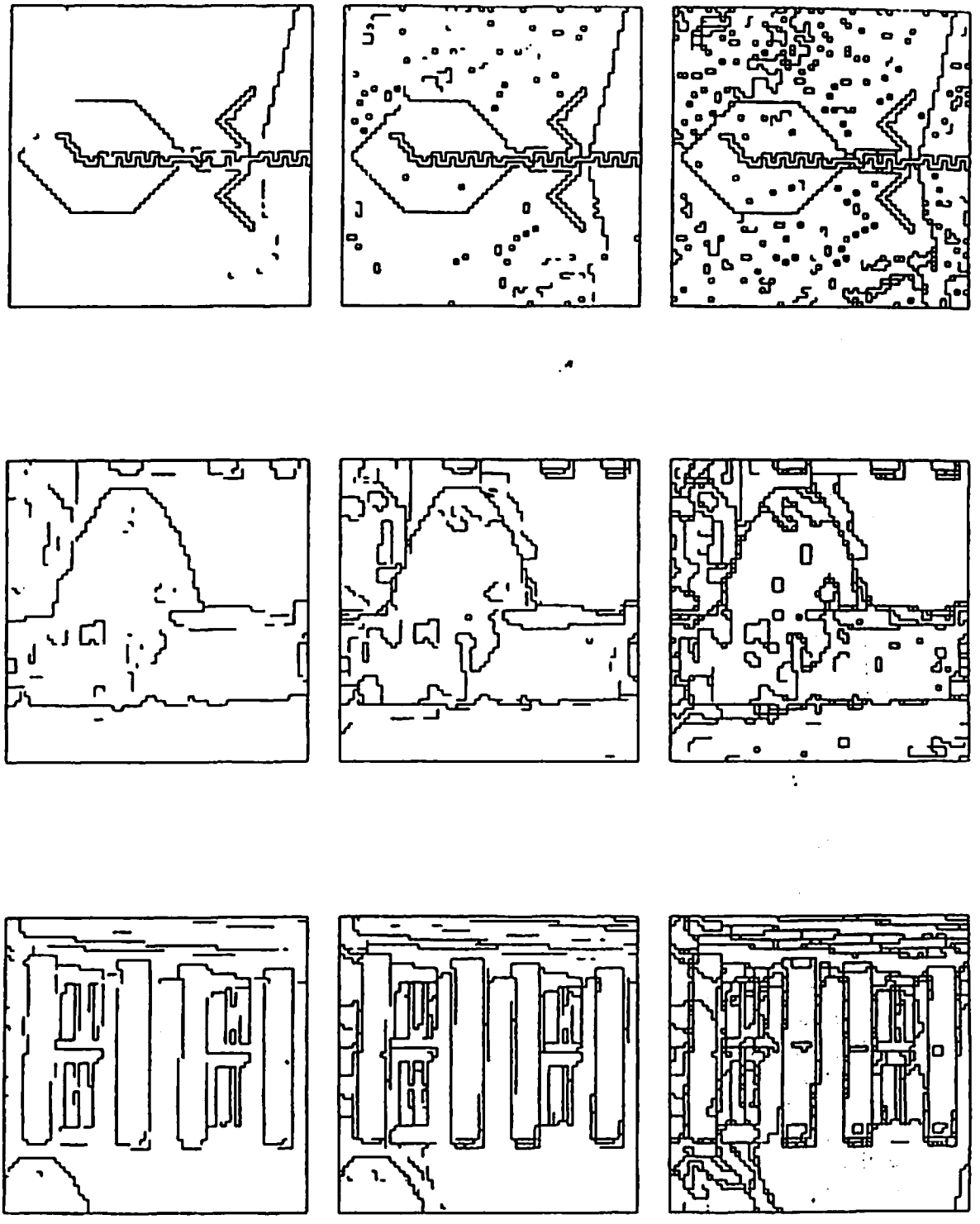


Figure 68: Overlayed Segmentations of Virus, Bush, and Window.

This artificial scene consists of two cylinders on a blue background. Figure 69 shows the image red, green, blue components, and intensity. Since the colors in this scene are fairly saturated and of similar brightness, segmentations based on intensity alone are virtually useless.

Each of figures 70 - 72 shows the raw red, green, and blue intensity images from left to right in the top row. The second row shows the corresponding segmentations produced by the region clustering algorithm. The third row shows the corresponding segmentations computed using the multi-threshold algorithm while the bottom row shows the corresponding segmentations computed using the edge relaxation algorithm.

Unified segmentations through merging.

By applying a region merging algorithm very similar to that described in chapter six to the union (the logical "or") of all of the possible segmentations, much of the undesirable oversegmentation is removed. The merge algorithm is the same as that used in section 6.7 with a straightforward extension to color images, e.g. using the euclidean color space distance as a merge factor in place of intensity difference. The algorithm was also extended to include additional rules for detecting "mixed pixel" regions and for blocking merges based on texture measure differences. Figure 73 shows the segmentations produced by this method for the four images used in section 6.1.2. The left column shows the segmentations before any merging has taken place, while the right column shows the resulting segmentation after the final region merge process described in section 6.6.2. The reader should compare these segmentations to to those of figures 65, 66, 69, and 70.

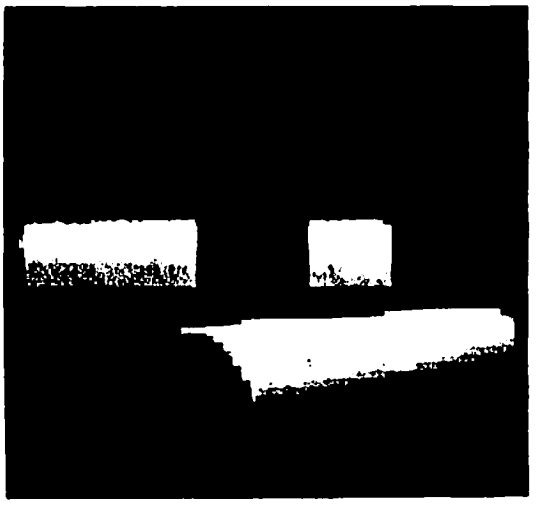
Combined segmentations by extending high confidence boundaries.

Another possible approach to combining the various segmentations begins with a core set of high confidence boundaries and extends this set with boundaries necessary for good continuation. Intuitively, we begin with boundaries which all or most of the the segmentation algorithms agree exist and iteratively add additional boundaries in order to form closed regions.[20]

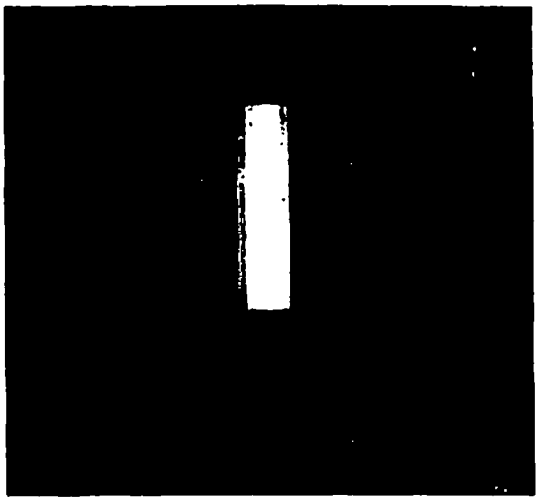
The first step in the algorithm is to select a set of high confidence boundaries from a candidate set of boundaries formed by the union of the boundaries in all the segmentations.[21] For each candidate boundary we compute a match score based on the average number of segmentations which agree on the boundary normalized

[20] This methodology is similar to the islands of reliability control strategy used in Hearsay II.

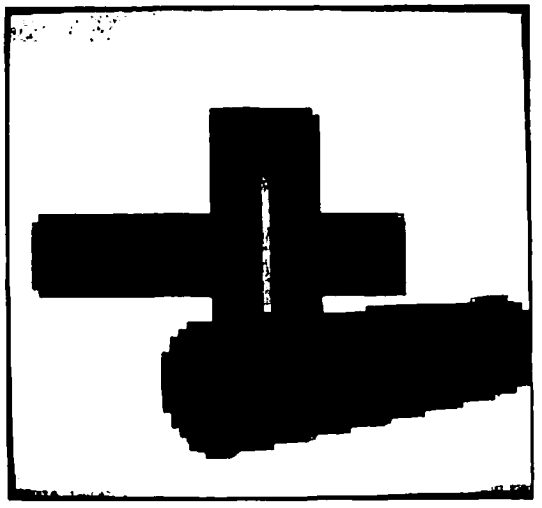
[21] By a boundary we mean a set of connected edge segments which contains no branch points (vertices). Boundaries are constrained to terminate at vertices of degree one, three, or four.



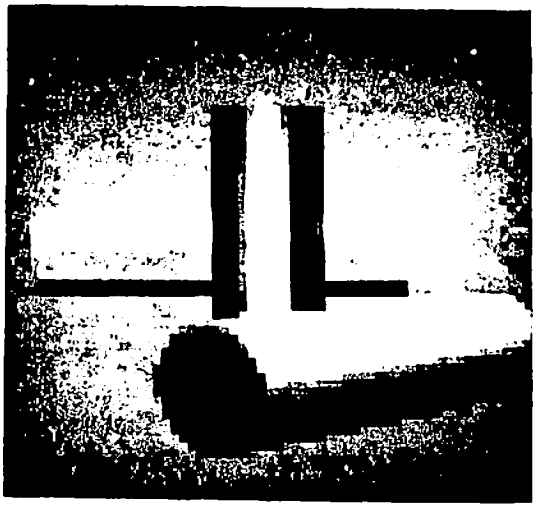
(a) Red



(b) Green



(c) Blue



(d) Intensity

Figure 69: Cylinder Scene.

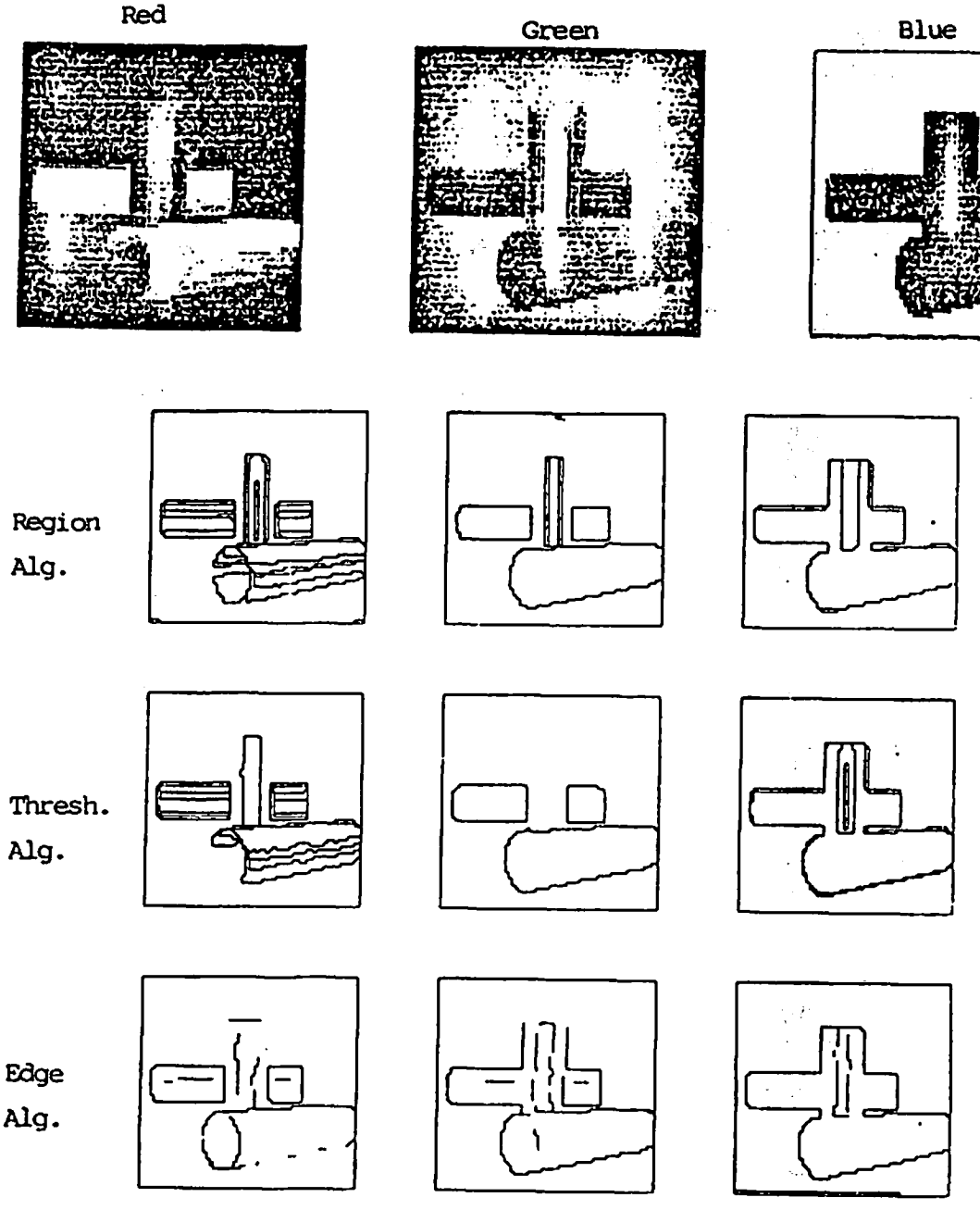


Figure 70: Cylinders - Segmentations by Color Components.

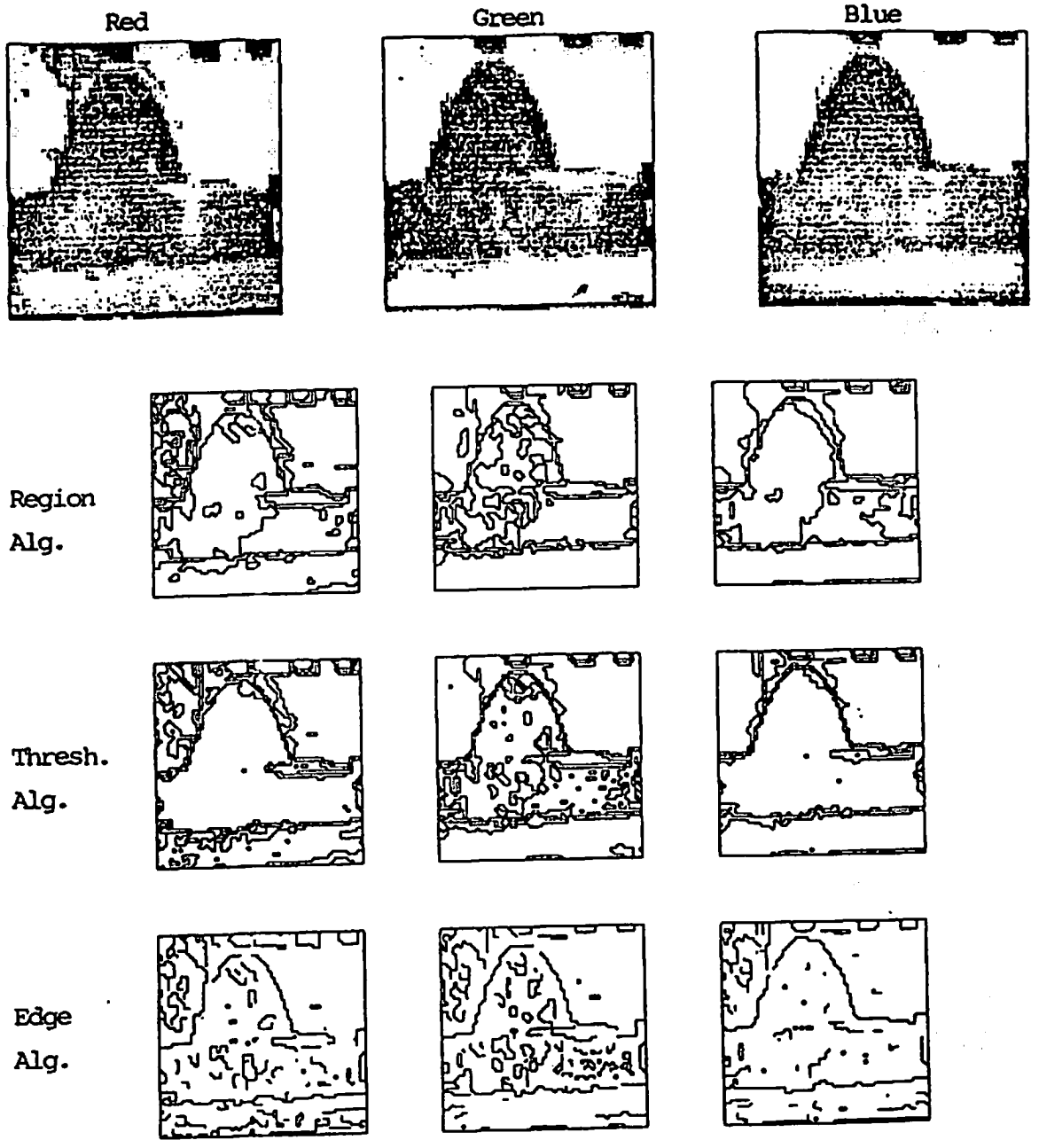


Figure 71: Bush - Segmentations by Color Components.

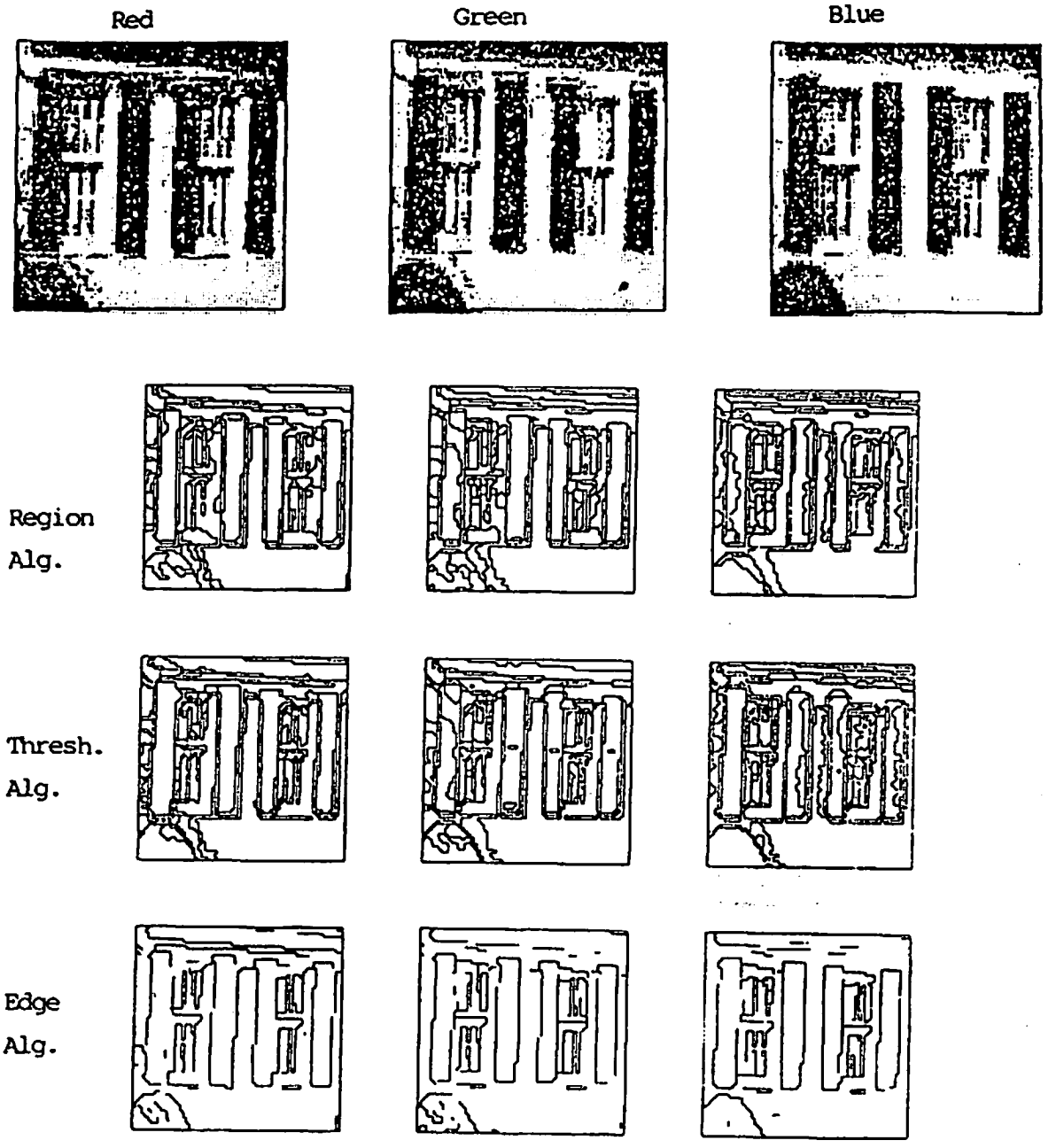


Figure 72: Window - Segmentations by Color Components.

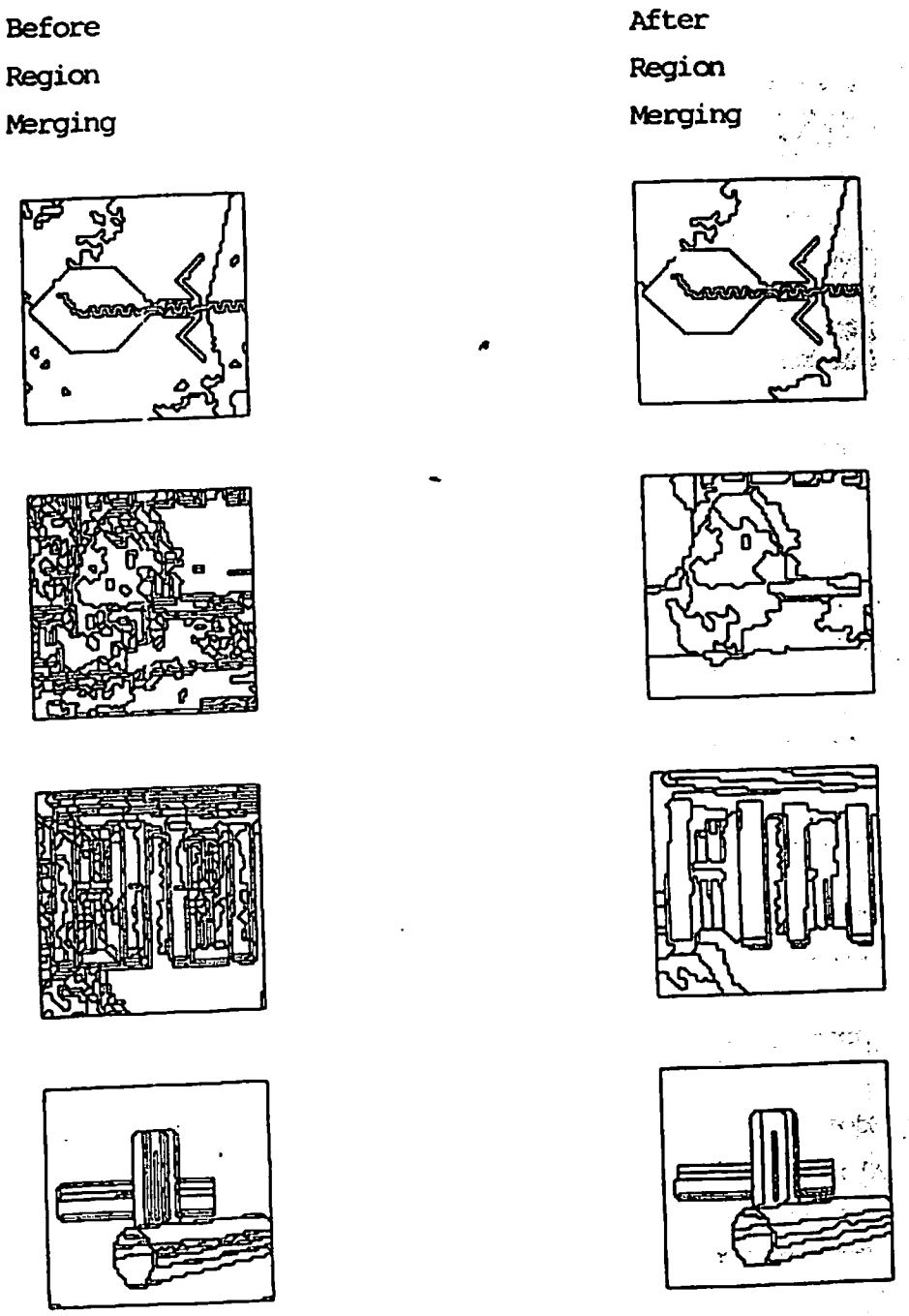


Figure 73: Remerging the Union of Alternative Segmentations.

by the number of alternative segmentations considered. A match score of 1.0 would mean that all of the segmentations agree on the placement of the entire boundary exactly. The match score would approach 1.0/(number-of-segmentations) when only one segmentation believes the boundary exists. The high confidence boundary set is computed by thresholding on the candidate set match scores. For all of the experiments below boundaries with match scores greater than .7 were accepted.

Once the initial high confidence boundaries are selected, the algorithm adds additional boundaries at degree one vertices. At each degree one vertex in the high confidence boundary set, the alternative boundaries from the candidate set which could extend this boundary are evaluated, and the boundary extension with highest match score is added to the current boundary set. If two alternatives have approximately equal match scores, then the shorter alternative boundary is selected. This commits the continuation to the shortest extension in these cases. Parallel hypothetical extension or backtracking are not considered here but could be of value in future extensions.

Figures 74, 75, 76, and 77 show the segmentation produced by this algorithm for the four images used in the previous section. The image in the upper left corner shows the initial candidate set of boundaries. The image in the upper right corner shows the initial high confidence boundaries selected by the first part of the algorithm. The lower left image shows the segmentation after extending all degree one vertices from the high confidence image iteratively. Note that any degree one vertices remaining were proposed by one of the edge relaxation segmentations and have no alternative extension in the candidate set. Finally, the image in the lower right shows the segmentation after region labelling and suppression of regions containing only one or two pixels.

The segmentations produced by this algorithm are quite good. However, the segmentations produced by the current algorithm may occasionally miss major image boundaries when no component boundary is included in the initial candidate set, and the algorithm may not form closed regions when the extension algorithm selects an extension to a boundary which is in fact an alternative positioning of the same boundary being extended. The worst segmentation was that of figure 77. The quality of this segmentation improves markedly if a match threshold of .6 is used for the initial candidate set; however, this setting would result in oversegmentation in figures 74, 75, and 76.

A number of modifications of this algorithm were evaluated, including alternative methods for selecting the candidate set, use of inexact matching to obtain match scores, and a somewhat more sophisticated extension algorithm which favored boundaries connecting two degree one vertices. None of these modifications improved the quality of the final segmentations. There are two possible extensions of this algorithm which we have yet to explore. The first extension would replace the criteria for extending high confidence boundaries to use more complex functions such as those used in the remerging algorithm. It is clear that the primitive match

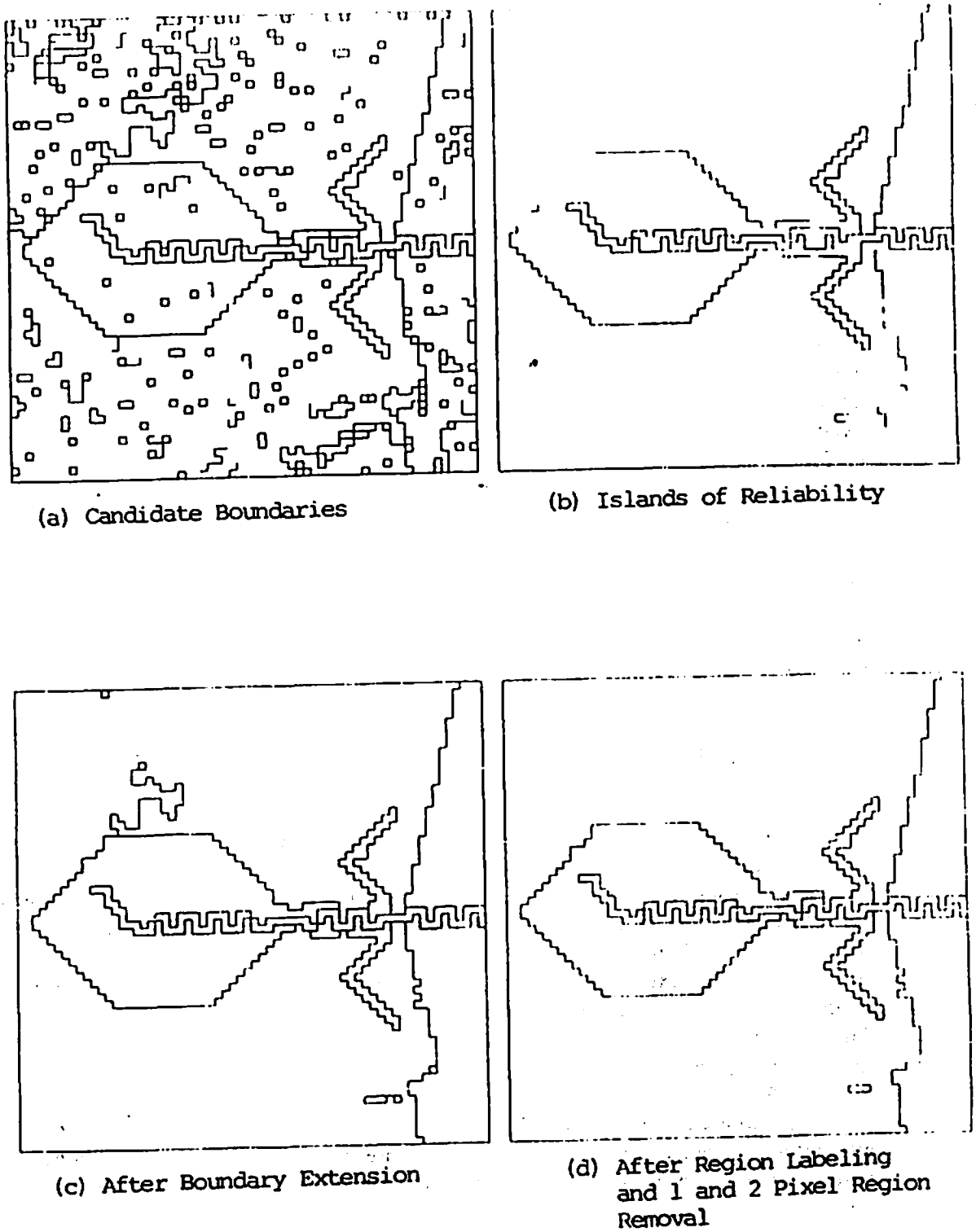
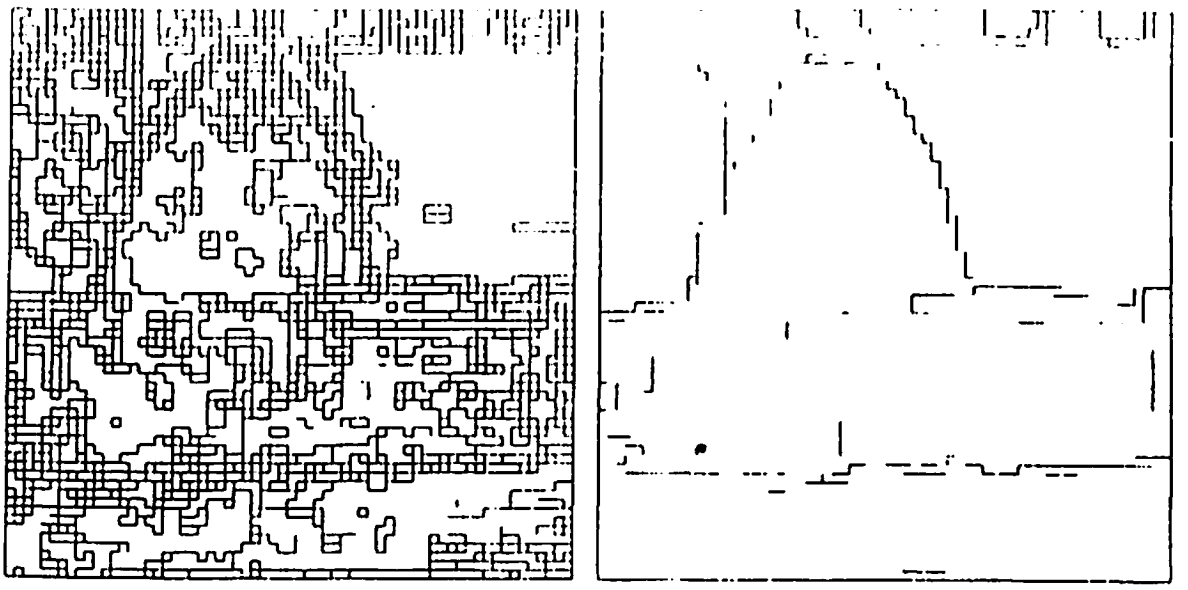
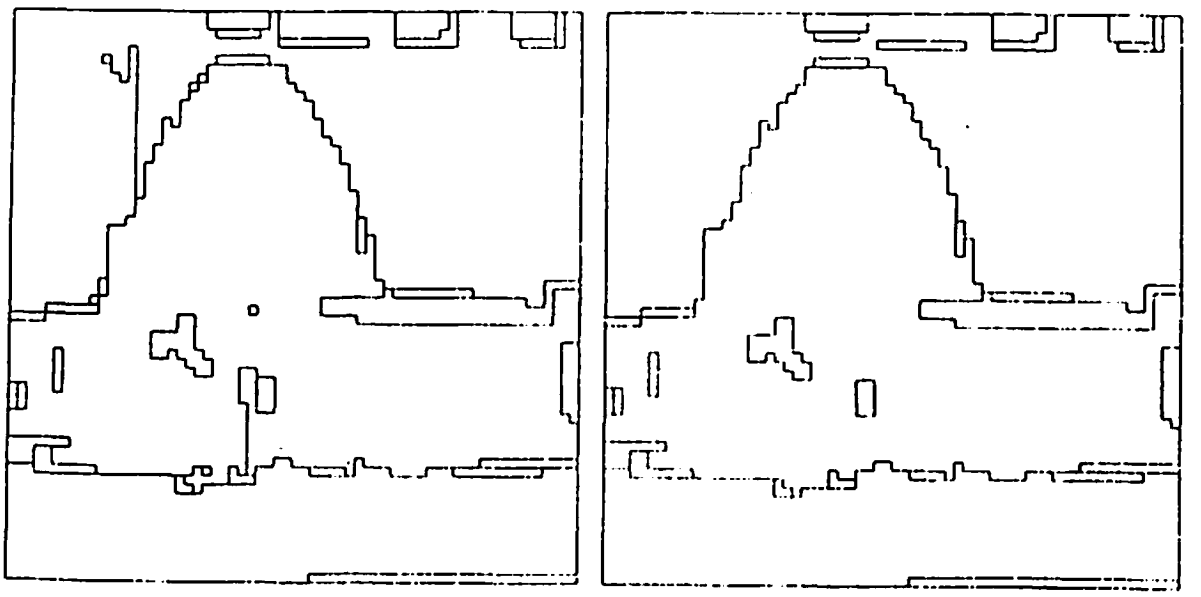


Figure 74: Virus - Boundary Closure Algorithm.



(a) Candidate Boundaries

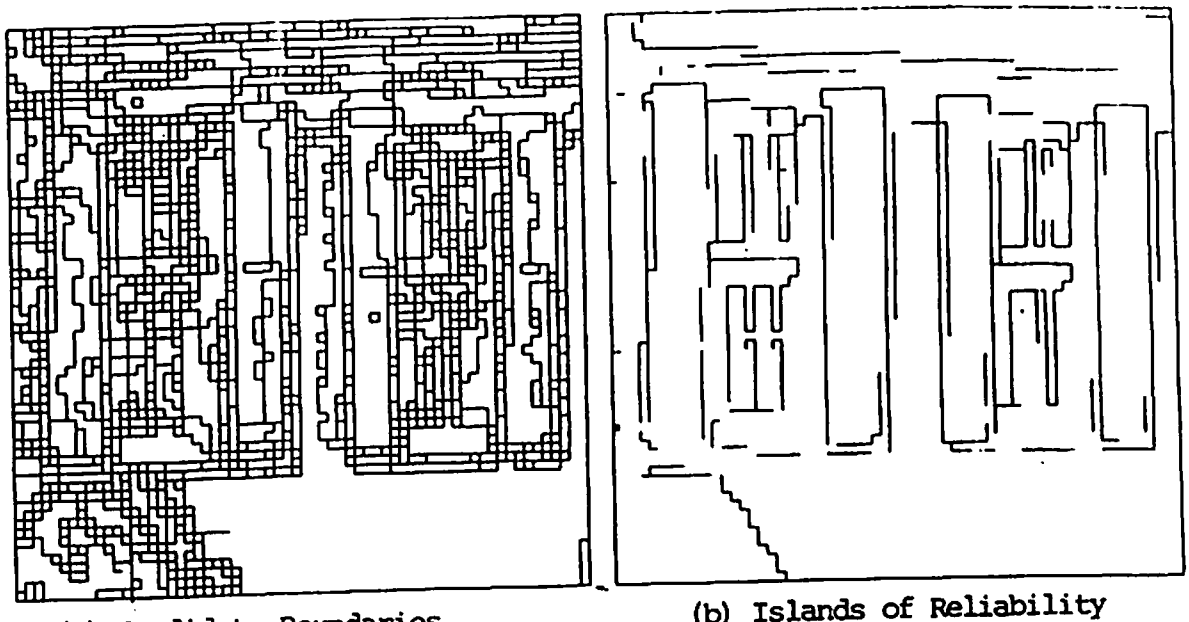
(b) Islands of Reliability



(c) After Boundary Extension

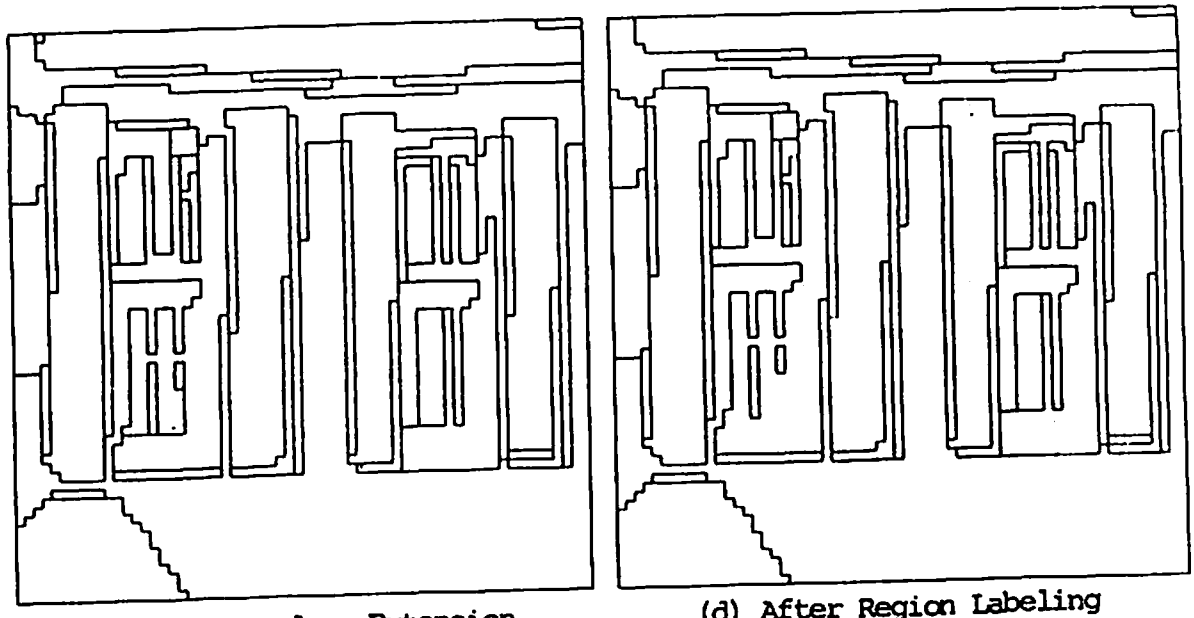
(d) After Region Labeling and 1 and 2 Pixel Region Removal

Figure 75: Bush - Boundary Closure Algorithm.



(a) Candidate Boundaries

(b) Islands of Reliability



(c) After Boundary Extension

(d) After Region Labeling and 1 and 2 Pixel Region Removal

Figure 76: Window - Boundary Closure Algorithm.

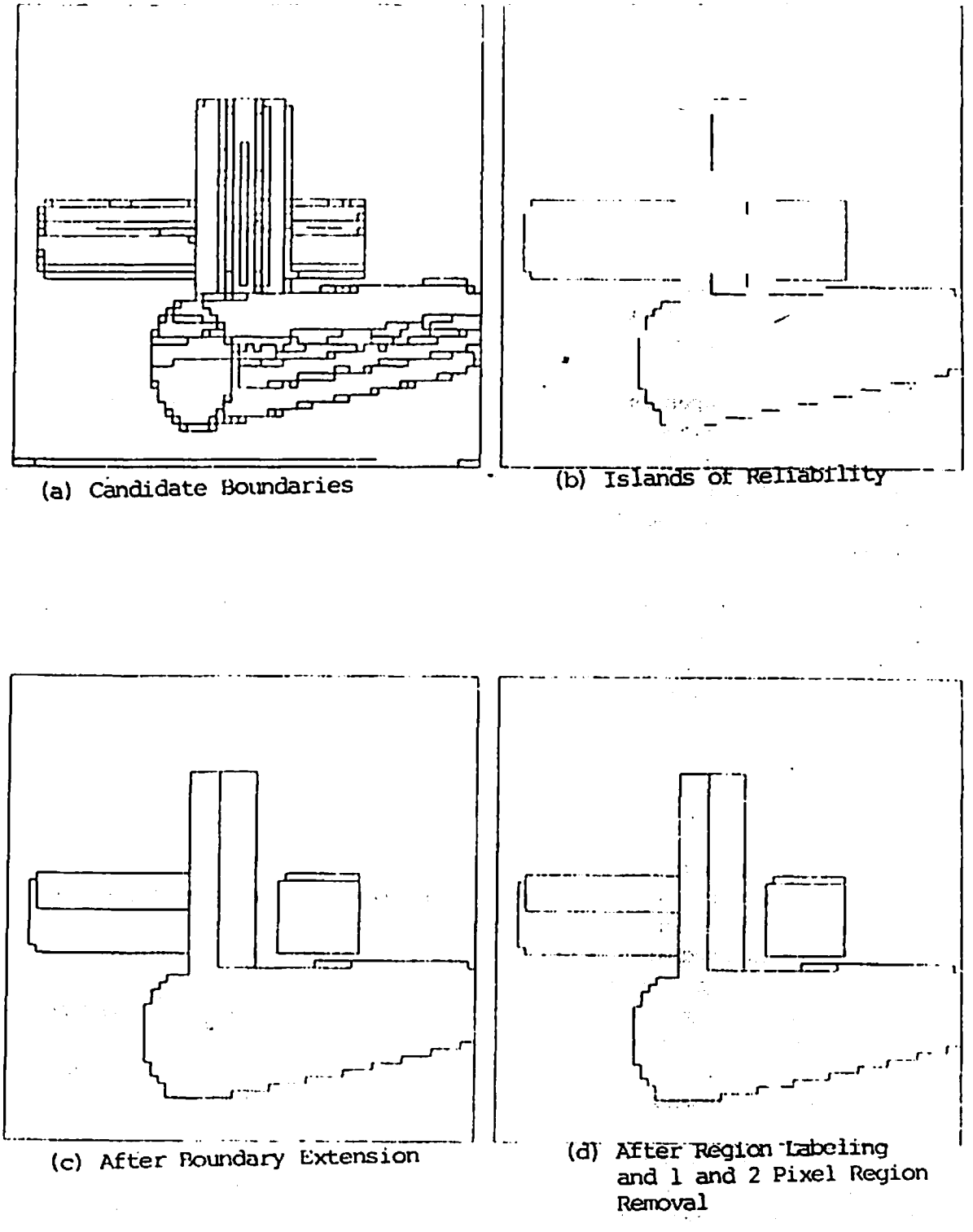


Figure 77: Cylinder - Boundary Closure Algorithm.

score used currently often leads to incorrect choices between alternative extensions. The second extension of the algorithm would replace the discrete selection of initial candidates and extension of boundaries with a relaxation algorithm which updated region boundary scores in much the same way as the edge relaxation algorithm of chapter III modified edge scores.

Another intriguing variant of this algorithm could locally weight each of the segmentations used to compute the match scores. The mechanism would be simply to point-wise multiply each segmentation with an image-dependent and algorithm-dependent weight image. This methodology could be used to increase sensitivity to boundary selection in areas where the semantic interpretation system expected a boundary but none was found, or to decrease sensitivity in areas of image texture for algorithms which tend to oversegment in the presence of texture. Furthermore, the methodology could change the relative contribution of the various segmentations to the match score. In particular, if one found that the threshold based algorithm was less sensitive to macro texture than the other two algorithms, one could locally increase the weight of the threshold algorithm boundaries in areas thought to contain macro texture, while simultaneously reducing the influence of the other algorithms in those areas by reducing their coefficients. One could similarly weight segmentations resulting from different features. If the goal of the segmentation process was to avoid fragmenting tree macro texture, one could increase the weight of the segmentations based on the green band over those based on the red band since the green spectral component is typically more uniform in tree areas than the red spectral component.

7.2 Dynamic Integration of Segmentations

One of the motivations behind the relaxation algorithms presented in chapter III is that it is best to postpone labeling decisions until as much evidence as possible has been obtained.[22] In these algorithms, evidence of cluster membership or edge presence was collected over a spatial neighborhood through the relaxation process. The cluster labeling algorithm knew nothing about possible edge placements advocated by the edge relaxation algorithm, and the edge relaxation algorithm knew nothing about the cluster labels on either side of an edge. We have argued that the edge and cluster algorithms operate on related but not identical information, and that it should be possible to integrate this knowledge to improve the segmentations.

In the case of the edge relaxation algorithm, the presence of an edge between two pixels is reinforced if the cluster algorithm finds that the pixels belong to different label classes. The presence of the edge is inhibited when the pixels have the same cluster affiliation probability distributions.

---

[22] Marr [MAR75] has called this the principle of least commitment.

Similarly, the compatability coefficients in the cluster relaxation scheme attempt to model local structure through a global measure. It could be argued that the edge information available from the edge relaxation algorithm might better represent the local geometry in the cluster afiliation updating than the compatibilities (or both factors could be used in conjunction). Pixels which are not separated by an edge should converge to the same label while pixels separated by an edge should converge to different labels.

In figure 78.a we see the parallel independent algorithms as they are described in chapter 3 while figure 78.b shows the integrated algorithm.

How the Edge Relaxation Algorithm uses Cluster Probabilities.

In the cluster relaxation algorithm the current labeling of each pixel is represented by a probability vector of cluster affiliations. We define the cluster algorithm's estimate of an edge between two adjacent pixels to be the Euclidian distance between the corresponding probability vectors. This measure is 1.0 if and only if both pixels have converged to different labels. The measure is 0.0 when both probability vectors are identical (and therefore it is likely that the pixels will converge to the same label).

As discussed in chapter 3, the edge relaxation update looks at all edges in a small neighborhood around the edge to be updated. In order to integrate the information from the region clustering algorithm, the edge algorithm simply views its neighborhood through a simple filter which takes a weighted mean of the edge currently estimated by the edge relaxation algorithm and the edge estimated by the cluster algorithm. In the experiments below both algorithms were weighted equally.

How the Cluster Algorithm uses Edge Probabilities.

The cluster relaxation algorithm updates a pixel in a neighborhood as stated in equations 3.2 and 3.3. We can integrate the edge information into the relaxation by replacing equation 3.2 with

$$q(i) = \sum_{x \in n} \left(1-P_x(e) \right) \sum_{j \in C} r_x(i,j) * P_x(j) \qquad 7.7$$

where $P_x(e)$ is the edge relaxation algorithm's estimate that an edge exists between the center pixel and neighbor pixel x. Effectively, this form implies that neighbors separated by high confidence edges do not contribute significantly to the update of the pixel. Since equation 3.3 is essentially a normalization, reducing the influence of one neighbor implicitly increases the relative contribution of the other neighbors.

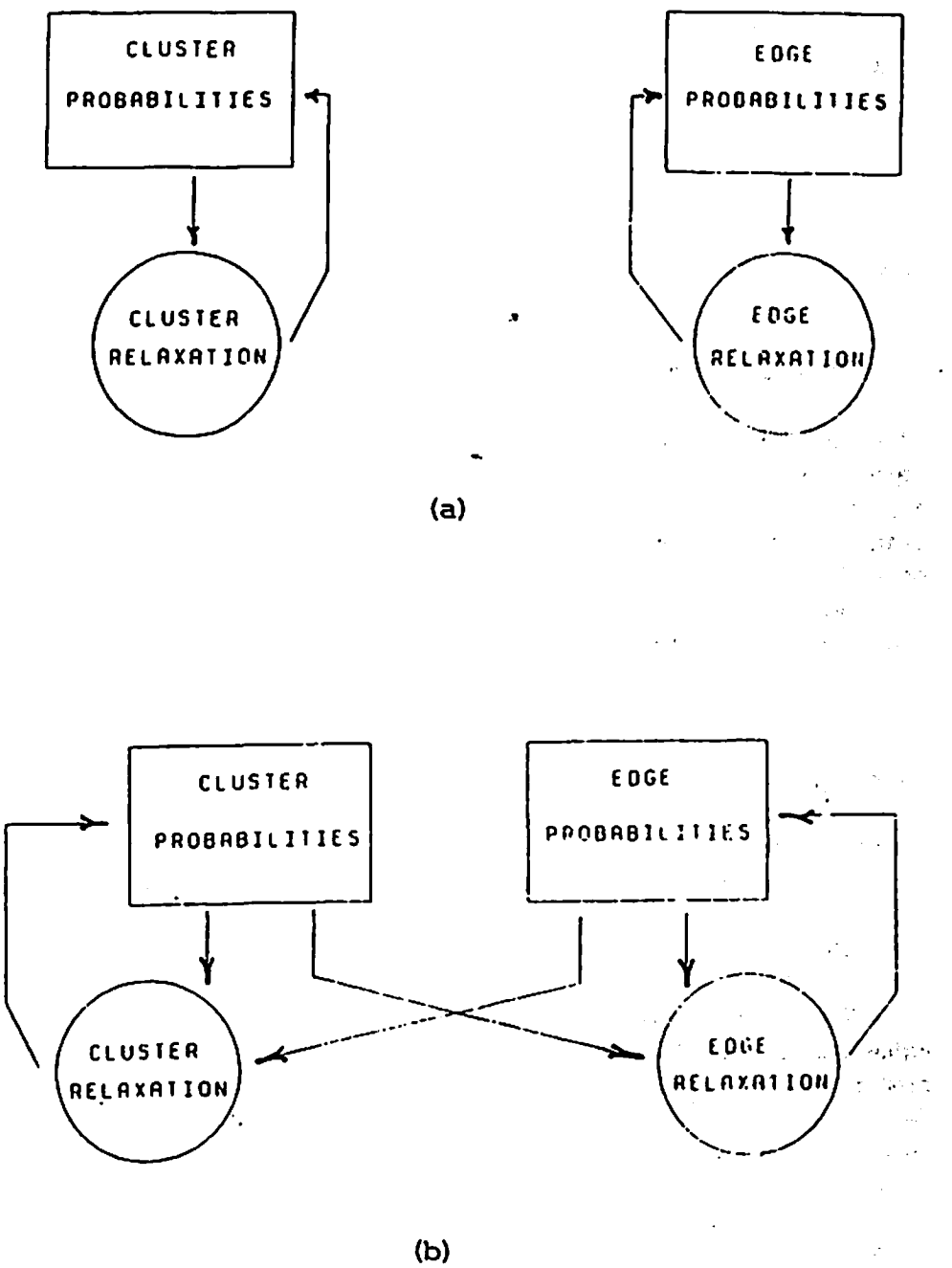


Figure 78: Model for Interacting Relaxation Processes.

Equation 7.7 integrates both the compatibilities and the edge probabilities in the update rule. In a second experiment we replaced the complex directional compatibilities based on the conditional probabilities of the initial pixel labeling with compatibilities where $r_\chi(i,j)$ is 1.0 when i equals j and -1.0 otherwise. It has been shown that use of these simple compatibilities will destroy fine image structure [NAG79], but use of the edge information should help to preserve those structures.

Interacting Relaxation Results.

In Figure 79 we can compare the interacting relaxation algorithm with the two independent algorithms. In this figure both the compatibility coefficients and the edges were used to update the cluster probability vectors. The left column shows the results of the independent global region cluster labeling relaxation algorithm run independently after 0, 2, 5 and 20 iterations of relaxation. The second column similarly shows the independent edge relaxation algorithm. The third and fourth columns show the results of the interacting cluster and edge algorithms, respectively. The third column shows the region cluster labeling algorithm which is using the edge algorithm information to update cluster affiliation probabilities. The fourth column shows the result of the edge algorithm which uses region algorithm cluster affiliation probabilities to update the edge probabilities. Figure 80 shows the same results when the cluster label compatibility coefficients are forced to 1 for like labels and -1 for different labels. Figures 81, 82, 83, and 84 show the corresponding results for two additional images

The effect of the edge algorithm on the cluster algorithm seems to be limited. The cluster algorithm is based on global clusters and even if the edge algorithm can decouple one or two pixels from the update, it cannot introduce new boundaries between regions which converge to the same global cluster. On the other hand, the cluster algorithm significantly reduces the number of edges in the edge algorithm segmentation which seem to correspond to noise.

The fine image structures which are destroyed by the relaxation with plus and minus one compatibility coefficients are preserved when the edge algorithm information is used as can be seen by comparing the columns one and three in figure 80. The area of the DNA in the virus image of figure 79 also indicates that the use of the edge information may better preserve these fine image structures than the compatibility coefficients alone.

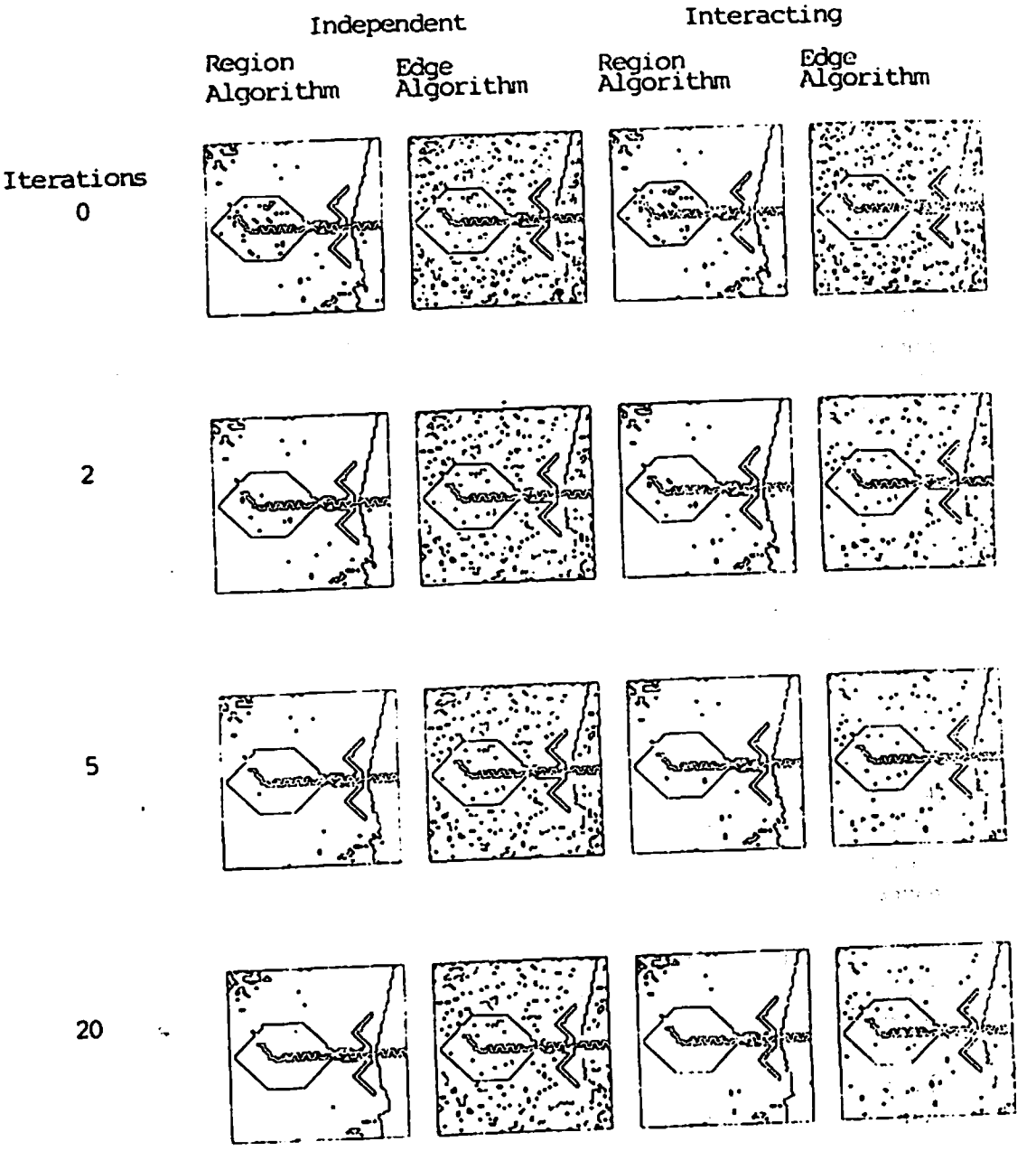


Figure 79: Interacting Relaxation - Virus with Regular Compatibilities.

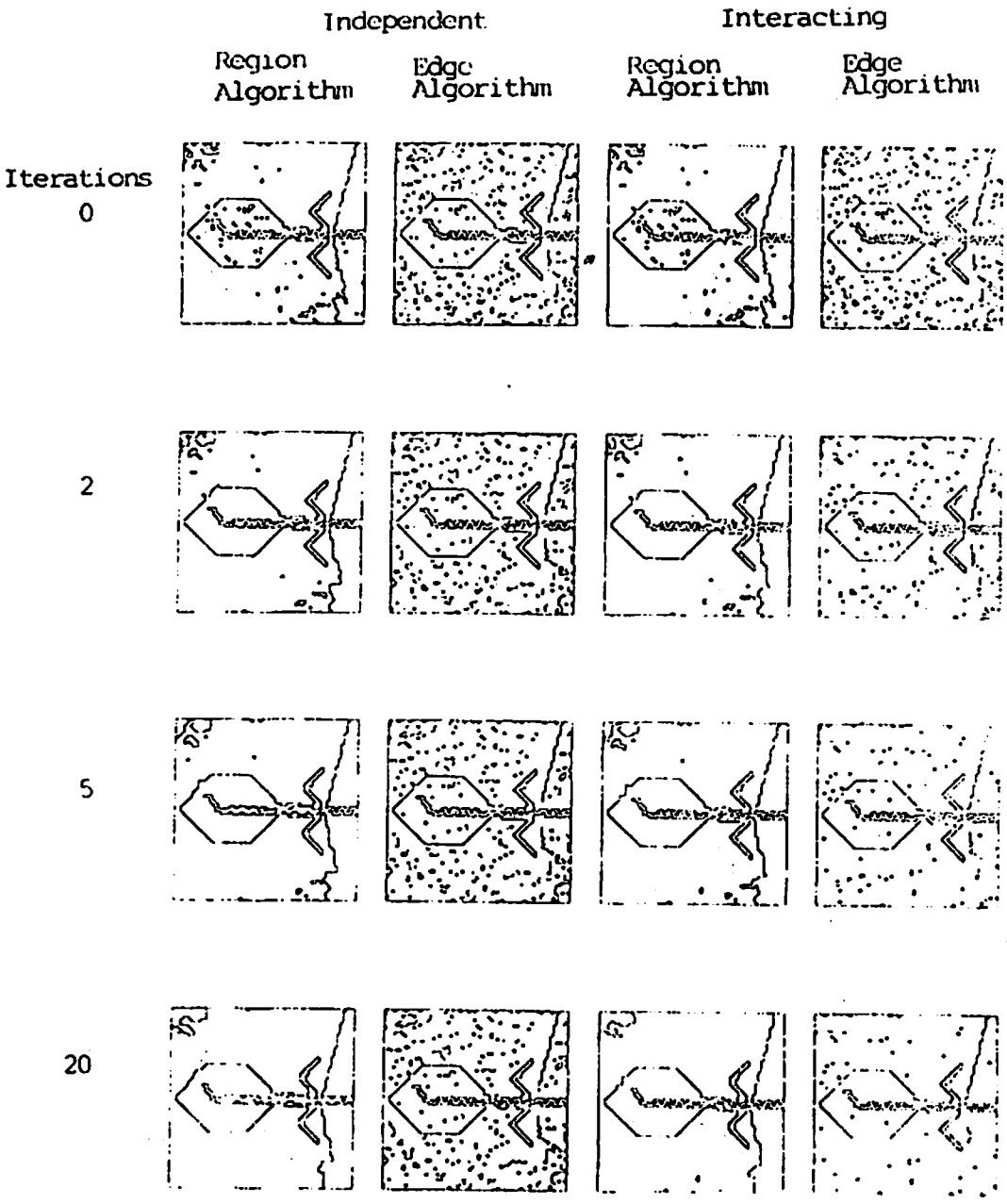


Figure 80: Interacting Relaxation - Virus with 0/1 Compatibilities.

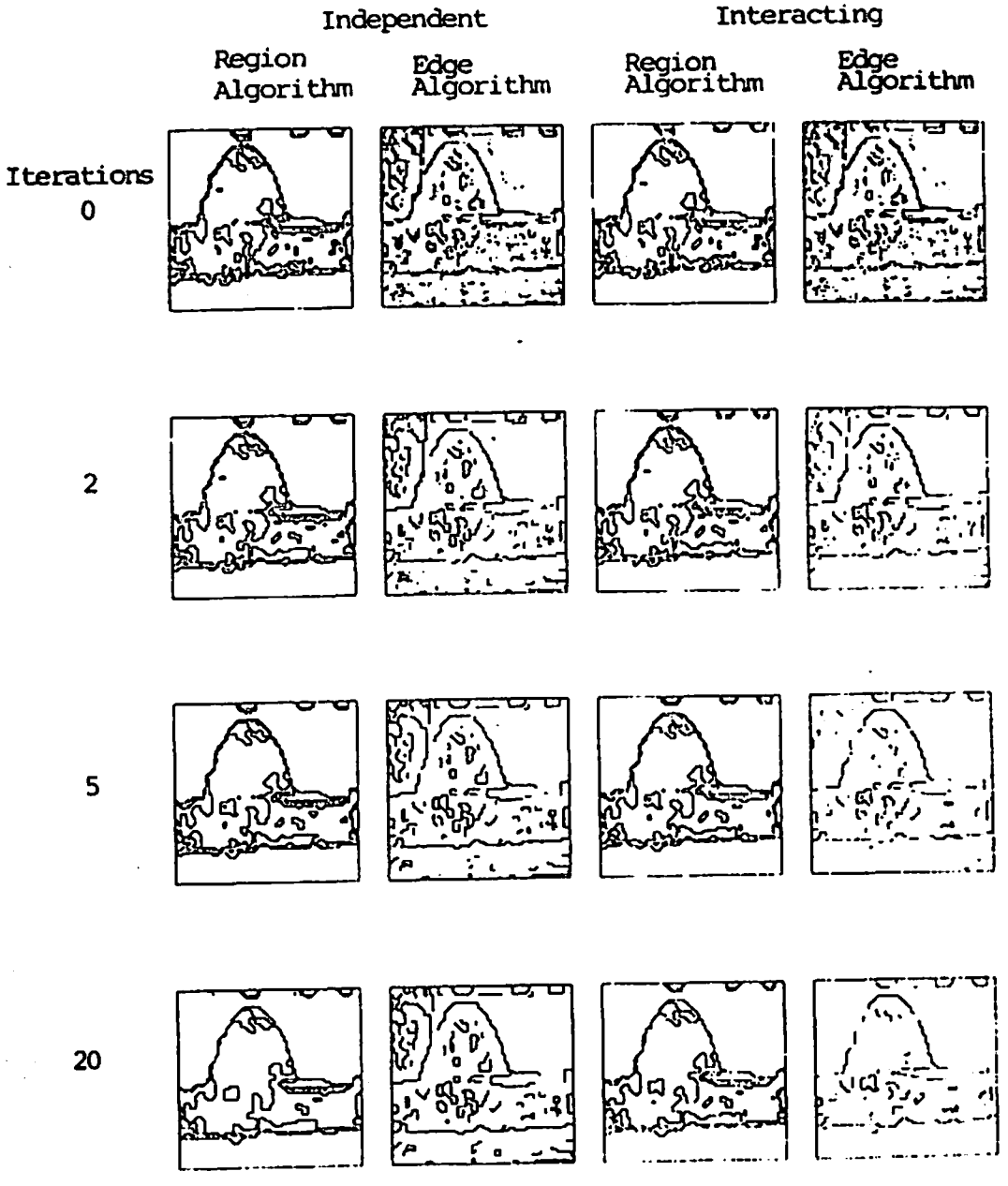


Figure 81: Interacting Relaxation - Bush with Regular Compatibilities.

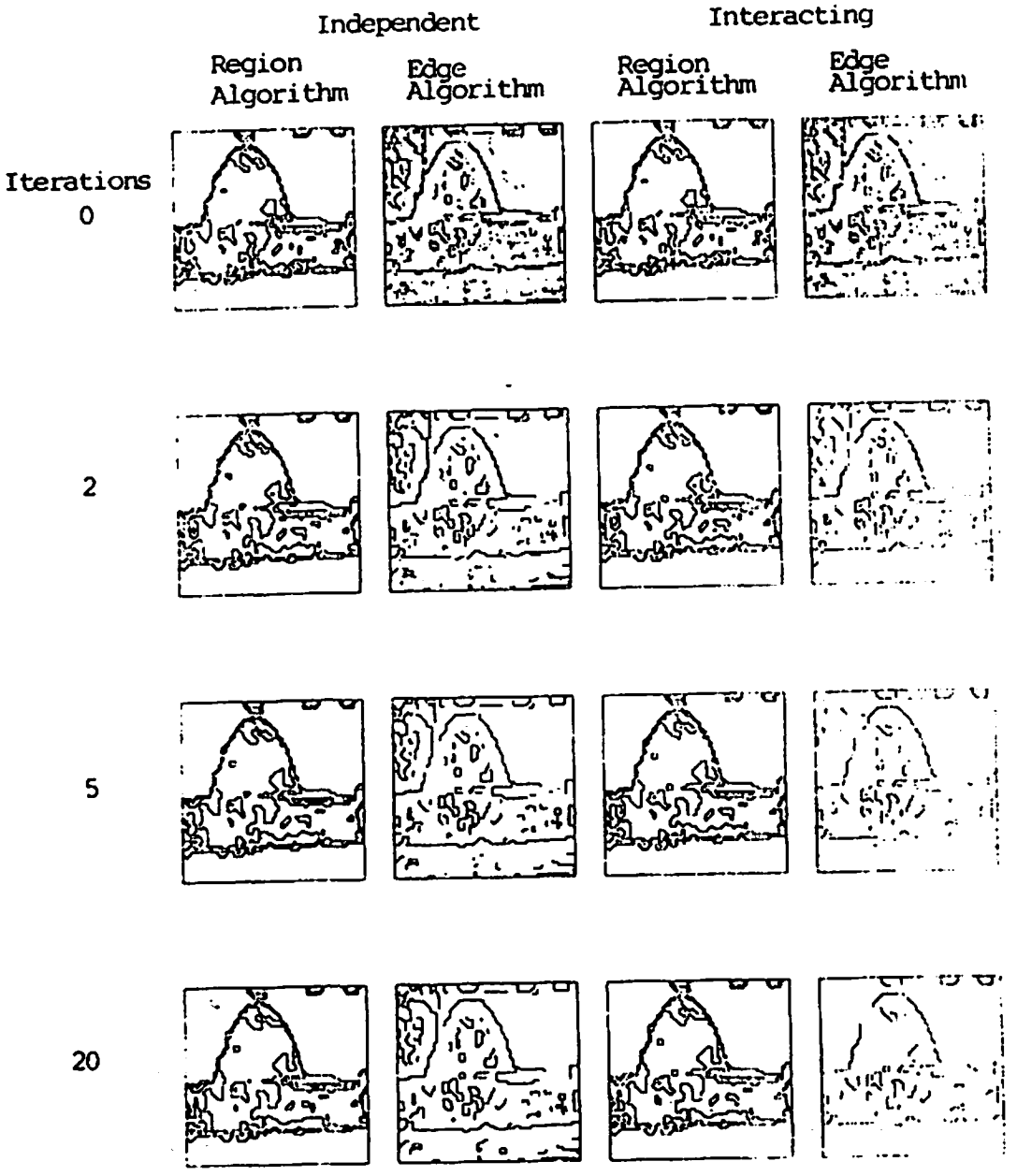


Figure 82: Interacting Relaxation - Bush with 0/1 Compatibilities.

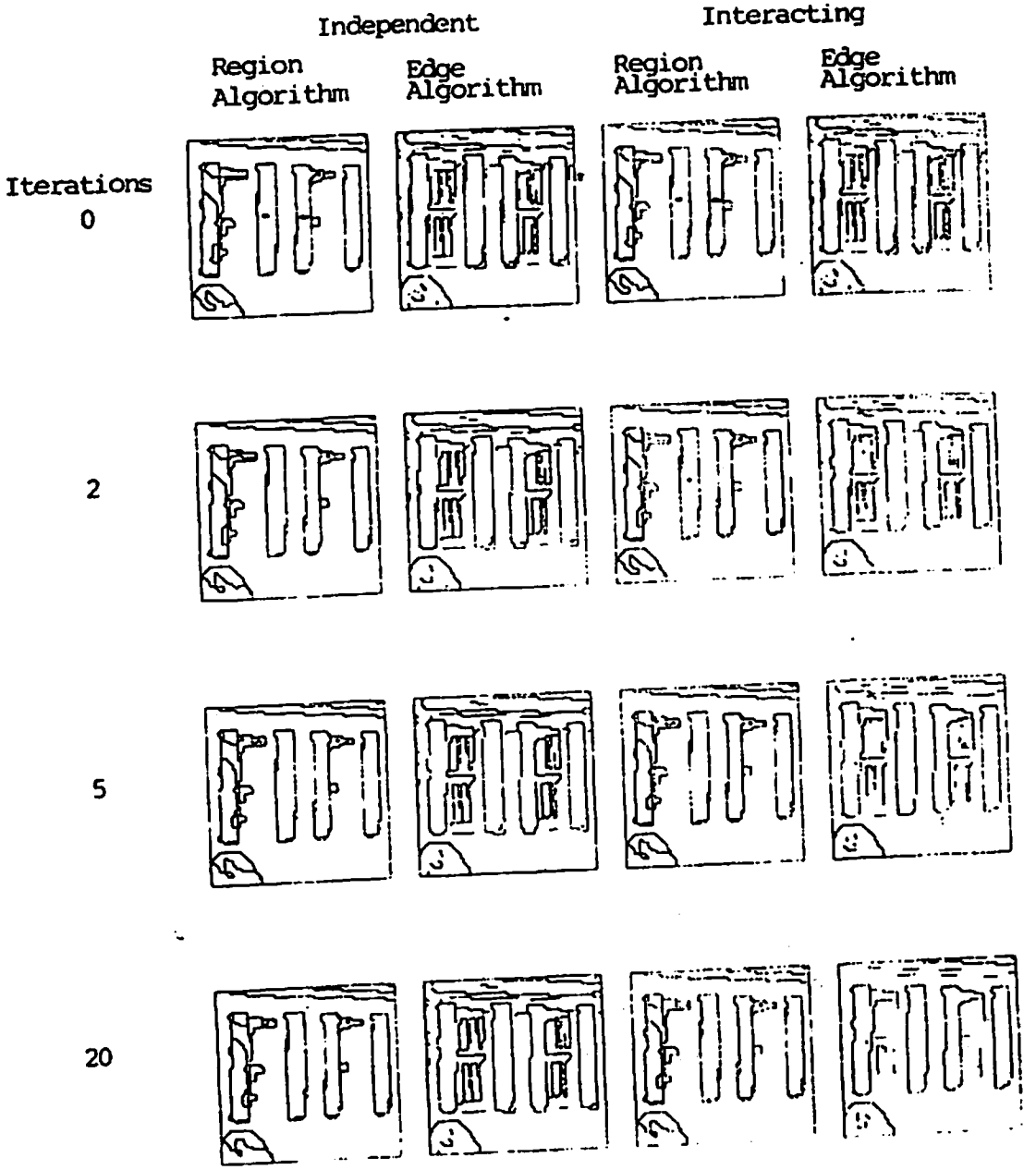


Figure 83: Interacting Relaxation - Window with Regular Compatibilities.

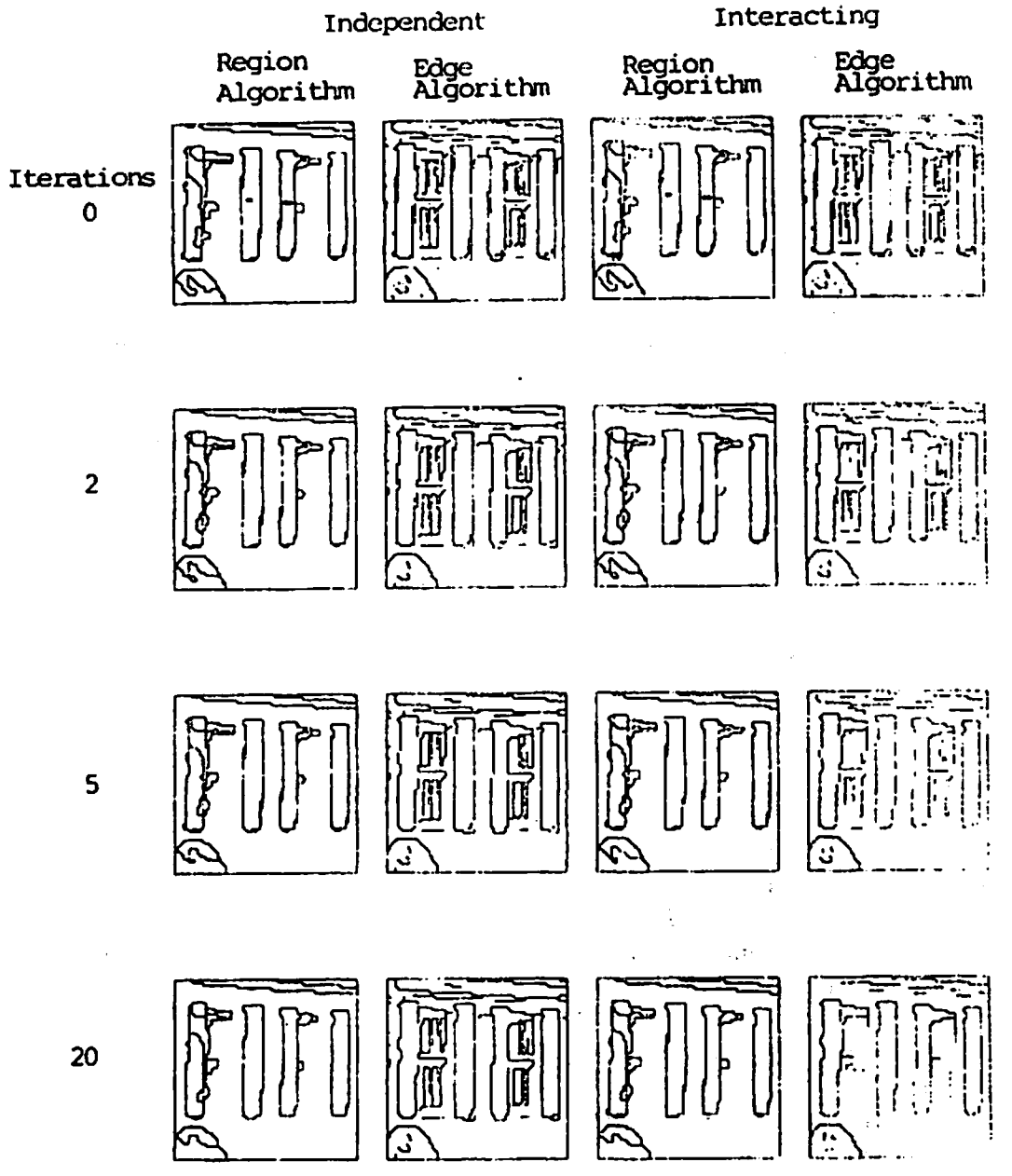


Figure 84: Interacting Relaxation - Window with 0/1 Compatibilities.

8.0 SUMMARY, FUTURE WORK, AND CONCLUSIONS

This chapter summarizes the contribution of the dissertation, discusses some of the major findings and issues, and proposes possible future work.

Our expressed goal was to produce initial segmentations of complex images with sufficient detail to initiate the interpretation process of the VISIONS system. This involves a tradeoff between the conflicting goals of producing a segmentation that faithfully represents image detail and the computational considerations that suggest a small number of regions is desirable. Since natural scenes often contain small objects with fine geometric structure, intensity gradients, shadows, highlights, and texture, the calculation of reasonable segmentations for natural scenes is extremely difficult. Because most segmentation algorithms are based on a small set of heuristics which reflect both implicit and explicit assumptions about the image, no segmentation algorithm can be expected to produce good segmentations when the assumptions on which it is based are violated.

In order to overcome these difficulties, we have successfully developed segmentation techniques which integrate various types of non-semantic knowledge into the segmentation process such that this knowledge can be used when and where it is appropriate.

The dissertation began by presenting the "Visions Image Operating System", the integrated software environment which made all of the experiments in this dissertation possible. The IOS provides a powerful experimental environment, built on LISP, in which complex image analysis algorithms can be easily integrated and applied to images of different structure and resolution. The IOS is currently being used by many researchers in image analysis both at the University of Massachusetts and at several other sites. We believe the IOS to be a significant advance in software available for low level image processing because of its functionality and because it encourages the design of parallel algorithms which will run on future parallel computers. The IOS allows the reseacher to concentrate on his algorithm rather than on programming details by providing image access fuctions in the image operators that handle varied data types and image boundary conditions. The IOS LISP interface allows the researcher to dynamically experiment and build image analysis algorithms by composing sequences of simple image operators. The IOS strikes an effective balance between the need for dynamic experimentation in an image analysis research environment and the computational needs inherent in the image processing domain.

In chapter 4 we extended a region clustering algorithm proposed by Nagin in two ways. First we showed that neither a minimum distance classifier nor a valley based decision boundary for cluster formation could be shown to be effective for all images. We developed a heuristic which balanced these approaches to produce a decision boundary which resulted in fewer post-relaxation errors. The second extension of the Nagin algorithm modified the center pixel compatibilities (which

determine the contribution of the center pixel in the relaxation update process) based on the goals of relaxation process. This technique not only lets us predict the behavior of the algorithm at local convergence, even in the multilabel case, but also lets us control the geometries which the relaxation will and will not preserve for any given image.

In chapter 5 we discussed the issue of segmentation evaluation. We provide no solutions to the perplexing problem of evaluating segmentations of complex outdoor scenes, but we do offer a methodology for segmentation algorithm development using images of increasing complexity, first to quantitatively and then to qualitatively evaluate the segmentations. Since the most advanced segmentation algorithms currently still make gross errors in segmenting images with texture, intensity gradients, and other complicating characteristics, we believe that subtle quantitative evaluation is not the most important problem to solve in the field of image segmentation.

In chapter 6 we investigated the integration of non-semantic knowledge into a single segmentation algorithm. We began by extending a simple segmentation algorithm which labels pixels with the feature histogram cluster to which they correspond within a relaxation labeling paradigm. This algorithm has a tendency to undersegment by failing to find clusters corresponding to small objects, and to oversegment by splitting intensity gradients into multiple clusters, by finding clusters for "mixed pixel" regions, and by finding clusters corresponding to microtexture elements. Furthermore, the relaxation process often destroys fine structure in the image. Nagin addressed the problem of missing clusters by localizing the process in the image space, which introduced the problem of inconsistent cluster sets in the artificial subimage partitions and the problem of combining the segmentations of the separate submimages into a consistent whole. Each of these problems was dealt with by adding and deleting clusters based on image space information, by merging regions, and by defining different compatibility coefficients in the relaxation so as to preserve fine structures. The result was a segmentation algorithm which more reliably produced better segmentations over a broader range of images than the simple clustering algorithm. We believe the resulting algorithm to be a state of the art segmentation algorithm which produces segmentations competitive with any existing segmentation algorithms.

The segmentations produced by this algorithm did raise questions as to the value of the cluster labeling relaxation processing since the relaxation was very computationally expensive but contributed very little to the final segmentations. It seemed that small region suppression (one and two pixels) achieved the majority of the effect of the relaxation. Although relaxation algorithms have tremendous intuitive appeal, their usefulness in our current form of histogram cluster labelling to produce regions seems limited. Basically the relaxation algorithms are local optimization procedures. As with many such procedures the results are good when the initial guess is close to an optimum, but can be much worse otherwise. Furthermore, the local convergence is not tied to the original data directly, thus

allowing the algorithms to "hallucinate". Very often their myopic view through small local windows is too limited to make good decisions for a more global optimization. In the case of our cluster based relaxation algorithm the problem is even worse, since this algorithm optimizes relative to a set of clusters which may be incorrect. As we have seen the incorrect selection of clusters can produce very bad segmentations.

One extremely interesting algorithm developed in chapter 6 is the region merging post-processing, which promises to be a very flexible approach independent of the other work presented in this dissertation. This algorithm effectively applies a number of small "knowledge sources" or merge rules to the merging of regions much in the same way that expert systems operate. Each merge rule may be applicable or not to a given merge decision and if applicable the rule contributes a weighted vote either in favor of the merge or opposed to the merge. The weight of the vote is proportional to the power of the rule, the confidence that the rule is appropriate for a given situation, and the confidence of the merge/no-merge decision itself. Since the rules set can be arbitrarily augmented, and the influence of each rule can be dynamically modified, this algorithm can provide a powerful framework for a segmentation executive. The segmentation executive could make segmentation decisions, perhaps based in part on feedback from the semantic interpretation system, and affect those decisions by altering the region merge rule set and individual rule parameters, by focussing on interesting image areas, or by weighting the various image features used in merge decisions.

Finally, in chapter 7 we investigated the integration of several segmentation processes using different segmentation algorithms, where the algorithms used each are based upon different assumptions about the image, and therefore have different strengths and weaknesses. We considered two approaches: integrating the segmentation results after they were independently computed, and integrating the algorithms directly during the relaxation process. The static integration using merging seemed quite effective and the algorithm which extended boundaries outward from islands of reliability, while not as successful, could be developed further.

When we integrated the region and edge algorithms directly, we found that the edge algorithm significantly reduced the number of isolated "noise" edges remaining after the relaxation process compared to the edge algorithm which did not utilize region information. We found no significant differences between global cluster label relaxation algorithm using and not using edge information.

In these experiments it was also observed that if edge information is utilized by the cluster label relaxation algorithm, the rather complicated compatibilities in the cluster relaxation process might effectively be replaced by simple +1 or -1 compatibilities, without seriously degrading the algorithm's ability to preserve fine image structure. Thus, edge information has replaced conditional probabilities that had to be derived from the image structure.

8.1 Future Work

A number of straightforward extensions of the approaches taken in this dissertation have been mentioned throughout. Beyond these, the method of integrating knowledge to select parameters could be extended to other algorithms such as the edge relaxation algorithm. One could also apply the same approach to the selection of the image features to which the algorithm is applied. Ohlander [OHL75] and Coleman [COL79] both approached the problem of feature selection using only the feature histograms (one dimensional and multidimensional histograms respectively) ignoring the effect of the decisions in image space. Selection of image features utilizing both feature space and image space information represents a major research area for the future.

Hierarchical Approaches.

As mentioned in section 7.1.1, a serious problem evident in both relaxation algorithms used in this dissertation is the lack of adequate context. The algorithms often find locally consistent solutions that are not globally consistent or desirable. In particular, the cluster region algorithm may preserve two pixel regions since neither pixel, with only a three pixel by three pixel view of the world, is aware that it is not part of some larger region. In the edge relaxation algorithm boundaries often fail to complete correctly when a single pixel is surrounded by boundaries at the end of the segment. In such a case the local connectivity constraints are satisfied but more global continuity is lost. Although these problems could be partially addressed by extending the relaxation neighborhoods, a more promising solution would be to integrate hierarchical relaxation algorithms operating at different levels of resolution. This could be viewed as an extension to the interacting relaxation algorithm methodology introduced in chapter 7. The alternative segmentation processes would use the same algorithm but be applied to the same image at different levels of resolution.

Alternative Clustering Algorithms.

It would be desirable to develop new cluster selection algorithms which integrate the concept of localization without the arbitrary partitioning of the scene into rectangular sub-images. In chapter 3 we saw that some form of localization was critical in detecting clusters corresponding to small regions. The localization into rectangular subimages did make it possible to locate the smaller hard to find clusters, but differences between cluster sets in the independent subimages made the task of "sewing" the pieces back together very difficult. The cluster addition algorithm alleviated this problem by ensuring that all necessary clusters were represented in the neighboring image sector; however artifacts at the junction between boundaries sometimes remain, since the clusters may be slightly different in the adjacent subimages. We would propose an alternative to this fixed sector cluster selection, where cluster sets are estimated at each pixel and a hierarchical relaxation algorithm selects a locally consistent cluster set at each pixel. Such an algorithm

would combine the advantages of localization in cluster selection without the artificial partitioning problem. Furthermore, the algorithm would be able to form regions in an intensity gradient without fragmentation, since the cluster center could drift over the spatial domain of the region. Hierarchical filtering of cluster alternatives would probably be desirable here as well.

8.2 Conclusions.

Many of the current state of the art segmentation algorithms use only a few assumptions or heuristics about image characteristics to partion an image into regions. These algorithms have, for the most part, not succeeded in producing useful segmentation across a broad range of images from different domains. The algorithms developed in this dissertation have integrated various kinds of non-semantic knowledge into the segmentation process. We suggest that these extended algorithms produce effective segmentations more reliably than the algorithms upon which they were based.

REFERENCES

BAL78 Ballard, D.H., Brown, C.M. and Feldman, J.A. 1978. "An approach to knowledge-directed image analysis," *Computer Vision Systems*, Hanson, A.R. and Riseman, E.M. (eds.), New York: Academic Press.

BRI70 Brice, C.R. and Fennema, C.L., "Scene Analysis Using Regions," *Artificial Intelligence*, 1, pp 205-226, 1970.

BUR83 Burt, P., "The Pyramid as A Structure for Efficient Computation," *Multiresolution Image Processing and Analysis*, (Rosenfeld ed.), Springer Verlag, 1983.

CHO71 Chow, C.K. and Kaneko, T., "Boundary Detection of Radiographic Images by a Threshold Method," *Proc. IFIP 71*, Ta-7, 130, 1971.

COL79 Coleman, G.B. and Andrews, H.C. "Image Segmentation by Clustering," *Proc. of IEEE*, 67-5, pp 773-785, 1979.

DAV75 Davis, L.S. "A Survey of Edge Detection Techniques," *Computer Graphics and Image Processing*, 4, pp 248-270, 1975.

DUD73 Duda, R.O. and Hart P.E. *Pattern Classification and Scene Analysis*, New York: Wiley & Son, 1973.

DUF81 Duff, M.J.B. and Levialdi, S. *Languages and Architectures for Image Processing*, New York: Academic Press, 1981.

ERI78 Ehrich, R.W. and Foith J.P. "Topology and Semantics of Intensity Arrays," *Computer Vision Systems*, (Hanson and Riseman Ed.) Academic Press, pp 111-127, 1978.

ERM80 Erman, L., Hayes-Roth, F., Lesser, V. and Reddy, D., "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys*, 12(2), pp 213-253, 1980.

FIR72 Firshein, O. and Fishler, M.A., "Describing and Abstracting Pictoral Structures," *Pattern Recognition*, 3, pp 421-444, 1971.

GLA83 Glazer, F., Reynolds, G., Anandan, P., "Scene Matching by Hierarchical Correlation," *Proc. 1983 Conf. on Computer Vision and Pattern Recognition*, Arlington Va., pp 332-441, 1983.

GRI83 Griffith, J.S., Kohler R.R., Hanson, A.R., and Riseman, E.R, "Region Merging using Multiple Non-semantic Rules," COINS TR in preparation.

GUZ68 Guzman, A. "Decomposition of a Visual Scene into Three-Dimensional Bodies," *Proc. FJCC*, 33, pp 291-304, 1968.

HAN74 Hanson, A.R. and Riseman, E.M. 1974. "Preprocessing cones: a computational structure for scene analysis," Coins Tech. Report No. 74C-7, University of Massachusetts, Amherst MA.

HAN75 Hanson, A.R. and Riseman, E.M., "The Design of a Semantically Directed Vision Processor (Revised and Updated)," Coins Tech. Report No. 75C-1, U. of Ma., Amherst MA., 1975.

HAN78a Hanson, A.R. and Riseman, E.M. "Segmentation of Natural Scenes," *Computer Vision Systems*, (Hanson and Riseman Ed.) Academic Press, pp 129-163, 1978.

HAN78b Hanson, A.R. and Riseman, E.M. "VISIONS: A Computer System for Interpreting Scenes," *Computer Vision Systems*, (Hanson and Riseman Ed.) Academic Press, pp 303-333, 1978.

HAN80 Hanson, A.R. and Riseman, E.M., "Processing Cones: A computational structure for image analysis," *Strucured Computer Vision*, (Tanimoto, S. and Klinger, A. eds.), New York: Academic Press, 1980.

HAN80b Hanson, A.R. Riseman, E.M., and Glazer, F.C., "Edge Relaxation and Boundary Continuity," COINS TR 80-11, U. of Ma., Amherst Ma., 1980.

HAR79 Haralick, R.M. and Watson, L., "A Facet Model for Image Data," *Proc. Pattern Recog. and Image Processing*, Chicago, pp 489-497, 1979.

HAY74 Hayes, K.C., Jr., Shah, A.N. and Rosenfeld, A., "Texture coarseness: further experiments," *IEEE Trans. Systems, Man, and Cybernetics* Vol. 4, pp 467-472, 1974.

HOR77 Horn, B.K.P. "Understanding Image Intensities," *Artificial Intelligence*, 8, pp 201-231, 1977.

HOU62 Hough, P.V.C., "Method and Means for Recognizing Complex Patterns", U.S. Pattent 3069654, 1962.

KAT65 Katz, Y.H. "Pattern Recognition of meteorological satellite photography," *Proc. 3rd Symp. on Remote Sensing*, U. of Mi., pp 173-214, 1965.

KEL71 Kelly, M.D. "Edge detection in pictures by computer using planning," *Machine Intelligence 6*, (Michie, D. ed.), Edinburgh, Scotland: Edinburgh University Press. pp 379-409, 1971.

KLI76 Klinger, A. and Dyer, C.R., "Experiments on picture representations using regular decomposition," *Computer Graphics and IMage Processing* Vol. 5, No. 1, pp 68-105, 1965.

KOH78 Kohler, R.R., "An Image Segmentation System based on Thresholding", COINS Department, U. of Ma., TR 78-23, 1978.

KOH81 Kohler, R.R., "A Segmentation System Based on Thresholding," *Computer Graphics and Image Processing*, 15, pp 319-338, 1981.

KOH84 Kohler, R.R., "Image Operating System User's Guide," COINS TR in preparation, U. of Ma., Amherst Ma., 1984.

KRU80 Kruse, B., "System architecture for image analysis," *Structured Computer Vision*, (Tanimoto, S. and Klinger, A. eds.), New York: Academic Press, 1980.

LES77 Lesser, V.R. and Erman, L.D., "A Retrospective View of Hearsay-II Architecture," *Proc. IJCAI-5*, Cambridge Ma, 1977.

MAG81 Maggiolo-Schettini, A. 1981. "Comparing some high-level languages for image processing," *Languages and Architectures for Image Processing*. New York: Academic Press.

MAR75 Marr, D. "Early Processing of Visual Information," *Trans. Royal Soc. London 275 on Biological Sciences*, pp 483-524, 1976.

MIL78 Milgram, D.L., "Edge Point Linking Using Convergent Evidence," *Proc. Image Understanding Workshop* (Baumann, ed.), 1978.

MIL79 Milgram, D.L., Kahl, D.J., "Recursive Region Extraction," *Computer Graphics and Image Processing*, 9, pp 82-98, 1979.

NAG79 Nagin, P.A. "Studies in Image Segmentation Algorithms Based on Histogram Clustering and Relaxation," COINS TR 79-15 and Ph.D. Thesis, U. of Ma., Amherst, 1979.

NAZ83 Nazif, A.M., "A Rule-based Expert System for Image Segmentation," Ph.D. Dissertation, E.E. Dept., McGill U., 1983.

OHL75 Ohlander, R.B., "Analysis of Natural Scenes," Ph.D. Thesis, CMU, Pittsburgh Pa., 1975.

OVE79 Overton, K.J. and Weymouth, T.E., "A Noise Reducing Preprocessing Algorithm," *Proc. Pattern Recog. and Image Processing*, Chicago, pp 498-507, 1979.

PER80 Perkins, W.A., "Area Segmentaiton of Images using Edge Points," *IEEE PAMI*, 1-2(1), 1980.

PRA79 Prager, J.M., "Segmentation of Static and Dynamic Scenes," Ph.D. Thesis, COINS TR 79-7, U. of Ma., Amherst Ma., 1979.

PRA78 Pratt, W.K., *Digital Image Processing*, Wiley & Sons, New York, 1978.

PRE66 Prewitt, J.S.M., "The Analysis of Cell Images," *Annals New York Acad. Sci.*, 128, 1035-1053, 1966.

PRI77 Price. K. and Reddy, R., "Change detection and analysis in multispectral images," *Proc. Fifth Int. Joint Conf. on Artificial Intelligence*, Cambridge MA. pp 619-625, 1977.

RIC80 Richards, J.A., Langdgrebe, D.A., and Swain. P.H., "Overcoming Accuracy Deterioration in Pixel Relaxation Labeling," *5th ICPR*, Miami Beach Fl., 1980.

ROB63 Roberts, L.G., "Machine Perception of Three-dimensional Solids," *Optical and Electro-optical Processing of Information*, (Tippet Ed.), 159-197. MIT, Cambridge Ma., 1963.

ROS71 Rosenfeld, A. and Thurston, M., "Edge and curve detection for visual scene analysis," *IEEE Trans. Computers* Vol C-20, No. 5, pp 562-569, 1971.

ROS76a Rosenfeld, A. Hummel, R., and Zucker, S., "Scene Labeling by Relaxation Operators," *Trans. Syst. Man and Cybernetics*, 6, pp 420-433, 1976.

ROS76b Rosenfeld, A. and Kak A.C., *Digital Picture Processing*, Academic Press, New York, 1976.

ROS80a Rosenfeld, A. and Samet, H., "Region representation: boundary codes from quadtrees," *CACM* Vol. 23, No. 3, pp 171-179, 1980.

ROS80b Rosenfeld, A., "Picuture processing: 1981," *Computer Graphics and Image Processing*, 1982.

TAN75 Tanimoto, S.L. and Pavlidis, T., "A hierarchical data structure for picture processing," *Computer Graphics and Image Processing* Vol. 4, No. 2, pp 104-119, 1975.

TAN78 Tanimoto, S.L., "Regular hierarchical image and processing structures in machine vision," *Computer Vision Systems*, Hanson, A.R. and Riseman, E.M. (eds.), New York: Academic Press, 1978.

TAN81 Tanimoto, S. and Klinger, A., *Structured Computer Vision*, New York: Academic Press, 1981.

TEN76 Tennenbaum, J.M. and Barrow, H.G., "Experiments in Interpretation Guided Segmentation," *Artificial Intelligence*, 2, 216-231, 1973.

TSU73 Tsuji, S. and Tomita, F., "A Structural Analyszer for a Class of Textures," *Computer Graphics and Image Processing*, 2, pp 216-231, 1973.

UHR72 Uhr, L., "Layered 'recognition cone' networks that preprocess, classify, and describ," *IEEE Trans. Computers*, 21, pp 758-768, 1972.

WAL75 Waltz, D.L., "Understanding Line Drawings of Scenes with Shadows," *The Psychology of Computer Vision*, (Winston ed.), McGraw-Hill, New York, 1975.

WAT74 Watanabe, S., "An Automated Apparatus for Cancer Prescreening: CYBEST," *Computer Graphics and Image Processing*, 3, pp 350-358, 1974.

WEB79 Webster, D.W., "Layered Relaxation Network for Object Detection" *SPIE*, 205, pp 92-101, 1979.

WES78 Weska, J.S., "A Survey of Threshold Selection Techniques," *Computer Graphics and Image Processing*, 7, pp 259-265, 1978.

YAK73 "Boundary and Object Detection in Real World Images," *Journal of ACM*, 23(4), 599-618.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: COINS TR 84-04	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): INTEGRATING NON-SEMANTIC KNOWLEDGE INTO IMAGE SEGMENTATION PROCESSES	5. TYPE OF REPORT & PERIOD COVERED: INTERIM
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Ralf R. Kohler	8. CONTRACT OR GRANT NUMBER(s): ONR N00014-75-C-0459
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer and Information Science Department, University of Massachusetts, Amherst, Massachusetts 01003	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Arlington, Virginia 22217	12. REPORT DATE: March 1984
	13. NUMBER OF PAGES: 183
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is 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)

segmentation
image processing
scene analysis
image processing software
segmentation evaluation
cluster based segmentation
region analysis
relaxation algorithms
edge algorithms

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

This dissertation develops several techniques for automatically segmenting images into regions. The basic approach involves the integration of different types of non-semantic knowledge into the segmentation process such that the knowledge can be used when and where it is useful. These processes are intended to produce initial segmentations of complex images which are faithful with respect to fine image detail, balanced by a

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

computational need to limit the segmentations to a fairly small number of regions.

Natural scenes often contain intensity gradients, shadows, highlights, texture, and small objects with fine geometric structure, all of which make the calculation and evaluation of reasonable segmentations for natural scenes extremely difficult. The approach taken by this dissertation is to integrate specialized knowledge into the segmentation process for each kind of image event that can be shown to adversely affect the performance of the process.

At the center of our segmentation system is an algorithm which labels pixels in localized subimages with the feature histogram cluster to which they correspond, followed by a relaxation labeling process. However, this algorithm has a tendency to undersegment by failing to find clusters corresponding to small objects; it may also oversegment by splitting intensity gradients into multiple clusters, by finding clusters for "mixed pixel" regions, and by finding clusters corresponding to microtexture elements. In addition, the relaxation process often destroys fine structure in the image. Finally, the artificial subimage partitions introduce the problem of inconsistent cluster sets and the need to recombine the segmentations of the separate subimages into a consistent whole. This dissertation addresses each of these problems by adding and deleting clusters based on image space information, by merging regions, and by defining different compatibility coefficients in the relaxation so as to preserve fine structures. The result is a segmentation algorithm which is more reliable over a broader range of images than the simple clustering algorithm.

Solutions to the same segmentation problems were examined via the integration of different segmentation algorithms (including edge, region, and thresholding algorithms) to produce a consistent segmentation. Multi-process integration techniques varied from static integration of the final results generated by individual algorithms through dynamic integration of the processes themselves. The resulting unified segmentations from these approaches were generally better than segmentations produced by any of the constituent algorithms.

Finally, the dissertation also describes the Visions Image Operating System (IOS) which made all of the experiments in this dissertation possible. This software environment, driven by an interactive user interface in LISP, provides a powerful experimental tool in which complex image analysis algorithms can be easily integrated and applied to images of different structure and resolution. The IOS is currently being used by many image analysis researchers at the University of Massachusetts and at several other sites involved in industrial, remote sensing, and medical applications.