

A RULE-BASED EXPERT SYSTEM
FOR IMAGE SEGMENTATION

by

© Ahmed M. Nazif

B. Sc. (Hons.), M. Sc. Cairo University

A thesis submitted to the Faculty of Graduate
Studies and Research in partial fulfilment of
the requirements for the degree of
Doctor of philosophy.

Electrical Engineering Department
McGill University
Montreal, Quebec, Canada

March 1983

ABSTRACT

A major problem in robotics vision is the segmentation of images of natural scenes in order to understand their content. This thesis presents a new solution to the image segmentation problem that is based on the design of a rule-based expert system. General knowledge about low level properties of an image is formulated into production rules. A number of processes employ the rules to segment the image into uniform regions and connected lines. In addition to the knowledge rules, a set of control rules are also employed. These include meta-rules that embody inferences about the order in which the knowledge rules are matched. They also include focus of attention rules that determine the path of processing within the image. A third set of rules contains the strategy rules which are data-driven inferences about the control rules. They dynamically modify the processing strategy. Different rule ordering and focus of attention strategies are selected according to a set of performance parameters. These measure the quality of the segmentation output at any point in time. Experiments with the knowledge rules resulted in an optimal set based on output quality and processing efficiency. Overall system performance is shown to be qualitatively and quantitatively superior to previous segmentation algorithms.

RÉSUMÉ

Un grand problème de la vision robotique provient de la difficulté de segmenter les images naturelles pour pouvoir en comprendre le contenu. Cette thèse propose un système expert pour segmenter les images. Les connaissances empiriques sur les caractéristiques de base sont représentées par des règles de production. L'image est segmentée en régions uniformes entourées de lignes par des processus employant ces règles. En plus des règles de connaissances, nous employons des règles de contrôle. Celles-ci comprennent des règles qui établissent l'ordre à suivre des règles de connaissances. En plus, il y a des règles régissant le foyer d'attention qui déterminent le cours du traitement de l'image. Une troisième catégorie de règles détermine la stratégie à employer. La stratégie est définie par des conclusions au sujet des règles de contrôle obtenues à partir des données. Ainsi la stratégie est modifiée continuellement par ces règles. L'ordre des règles et la stratégie du foyer d'attention sont définies par un ensemble de paramètres de performance. Ceux-ci mesurent la qualité de la segmentation à tout instant. Des expériences avec les règles de connaissances ont abouti à un ensemble optimal basé sur la qualité de la segmentation et l'efficacité du traitement de l'image. Il est démontré que le fonctionnement d'ensemble du système est qualitativement et quantitativement supérieur aux algorithmes de segmentation antérieurs.

ACKNOWLEDGEMENTS

I wish to express my deep appreciation and sincere thanks to Prof. M. D. Levine for the time and effort that went into countless hours of discussions. His continuous interest resulted in many constructive suggestions and corrections. I also acknowledge his critical review of this manuscript.

Prof. S.W. Zucker, Dr. S. Shaheen, and all the graduate students at the Computer Vision and Robotis Lab. provided a friendly research atmosphere and a productive working environment. Pierre Parent managed the computer system and graciously tolerated my excessive acquisition of disk space and cpu time. Thanks are also due to Pierre Lamoureux for translating the abstract.

The encouragement and patience of my wife Mona made this an endurable experience. My parents provided the motivation and the support for which I am eternally grateful.

This research was partially supported by the National Science and Engineering Research Council of Canada under Grant No. A-4156, and by the Department of Education, Province of Quebec under Grant No. CE-633.

TABLE OF CONTENTS

	Page
ABSTRACT	ii
RÉSUMÉ	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	xi
LIST OF TABLES	xvi

CHAPTER 1 : INTRODUCTION

1. 1. OBJECTIVES	1
1. 2. SYSTEM OVERVIEW	6
1. 3. ORIGINAL CONTRIBUTIONS	15
1. 4. THESIS DESCRIPTION	21

CHAPTER 2 : COMPUTER VISION AND RULE-BASED SYSTEMS

2. 1. BACKGROUND	25
2. 2. COMPUTER VISION SYSTEMS	27
2. 2. 1. Line Analysis	28
2. 2. 2. Region Analysis	34
2. 2. 3. The Vision Hierarchy	42

2.3. KNOWLEDGE-BASED SYSTEMS	48
2.3.1. Production Systems	49
2.3.2. Rule-Based Systems in AI	52
2.3.3. Other Knowledge-Based Systems	56
2.3.4. Control in the Rule-Based Paradigm	58
2.4. RULE-BASED SYSTEMS FOR COMPUTER VISION	64
2.4.1. Review of Rule-Based Vision Systems	64
2.4.2. The Rule-Based Approach to Segmentation	72

CHAPTER 3 : THE DATA STRUCTURE

3.1. INTRODUCTION	78
3.1.1. The Database	80
3.2. THE STATIC IMAGE DATA	82
3.2.1. The Input Image	82
3.2.2. Gradient Information	84
3.3. THE SEGMENTATION DATA	87
3.3.1. Regions	87
3.3.1.1. Descriptive Features	89
3.3.1.2. Spatial Features	94
3.3.2. Lines	96
3.3.2.1. Descriptive Features	99
3.3.2.2. Spatial Features	104
3.3.2.3. Line and Region Relations	110
3.3.3. Focus of Attention Areas	113
3.3.3.1. Types of Areas	114
3.3.3.2. Descriptive Features	116
3.4. THE OUTPUT	120
3.4.1. Measure of Segmentation	121

3. 5. SYSTEM INFORMATION AND STATISTICS	131
3. 5. 1. Processing Information	131
3. 5. 2. Rule Statistics and Priority	132

CHAPTER 4 : THE PROCESSING MODULES

4. 1. THE INITIALIZER	136
4. 1. 1. Initializing Regions	137
4. 1. 2. Initializing Lines	138
4. 2. THE REGION ANALYZER	142
4. 2. 1. Merging Regions	142
4. 2. 2. Splitting Regions	143
4. 2. 2. 1. Histogram Splitting	146
4. 2. 2. 2. Splitting at Lines	149
4. 3. THE LINE ANALYZER	152
4. 3. 1. Deleting Lines	152
4. 3. 2. Extending Lines	153
4. 3. 2. 1. Local Gradient Method	154
4. 3. 2. 2. Proximity Method	156
4. 3. 2. 3. Extensions Based on Regions	157
4. 3. 3. Merging Lines	159
4. 4. THE AREA ANALYZER	162
4. 4. 1. Generating New Areas	162
4. 4. 2. Maintaining Current Areas	164
4. 5. THE FOCUS OF ATTENTION MODULE	165
4. 5. 1. Focusing on Areas	167
4. 5. 2. Focusing on Regions	169
4. 5. 3. Focusing on Lines	170
4. 6. THE SUPERVISOR	171

CHAPTER 5 : KNOWLEDGE AND CONTROL

5.1. INTRODUCTION	174
5.2. THE STRUCTURE OF THE RULES	179
5.2.1. The Rule Conditions	180
5.2.2. Symbol Manipulation	188
5.2.3. Formulating a Condition	197
5.2.3.1. Region Features	198
5.2.3.2. Line Features	200
5.2.3.3. Area Features	201
5.2.3.4. Other Features in a Condition	202
5.2.4. The Rule Actions	203
5.2.4.1. Actions on Areas	205
5.2.4.2. Actions on Regions	205
5.2.4.3. Actions on Lines	205
5.2.4.4. Focus of Attention Actions	206
5.2.4.5. Supervisor Actions	206
5.3. THE KNOWLEDGE RULES	207
5.3.1. Region Analysis Rules	212
5.3.2. Line Analysis Rules	220
5.3.3. Area Analysis Rules	232
5.4. THE CONTROL RULES	238
5.4.1. Focus of Attention Rules	239
5.4.2. The Meta-Rules	244

CHAPTER 6 : THE SYSTEM STRATEGY

6. 1. INTRODUCTION	251
6. 2. STRATEGY DEFINITION	255
6. 2. 1. Data Selection Strategy	257
6. 2. 1. 1. Region Path Strategy	258
6. 2. 1. 2. Line Path Strategy	258
6. 2. 1. 3. Updating Strategy	259
6. 2. 2. Rule Priority	260
6. 2. 2. 1. Processor Priority	261
6. 2. 2. 2. Region Rule Priority	262
6. 2. 2. 3. Line Rule Priority	263
6. 2. 3. Dynamic Strategy Setting	264
6. 3. PERFORMANCE PARAMETERS	270
6. 3. 1. Region Uniformity	271
6. 3. 2. Region Contrast	275
6. 3. 3. Line Contrast	279
6. 3. 4. Line Connectivity	282
6. 3. 5. Texture Measures	284
6. 4. STRATEGY DETERMINATION	288
6. 4. 1. The Principles of Fuzzy Logic	289
6. 4. 2. Global Strategy Evaluation	295
6. 4. 3. Local Strategy Evaluation	300
6. 5. CONCLUSIONS	310

CHAPTER 7 : RESULTS AND CONCLUSIONS

7. 1. SYSTEM IMPLEMENTATION	314
7. 2. KNOWLEDGE BASE EXPERIMENTS	317
7. 2. 1. Region Merging Experiments	322

7.2.2. Region Splitting Experiments	328
7.3.3. Initial Region Size Experiments	332
7.2.4. Line Deletion Experiments	336
7.2.5. Line Extension Experiments	342
7.2.6. Line Merging Experiments	344
7.2.7. The Effect of Lines on Regions	349
7.2.8. Knowledge Rules Evaluation	353
7.3. CONTROL STRATEGY EXPERIMENTS	355
7.3.4. Region Path Strategy Experiments	356
7.3.5. Line Path Strategy Experiments	362
7.3.6. Updating Strategy Experiments	368
7.3.3. Process Priority Experiments	374
7.3.1. Region Rules Priority Experiments	377
7.3.2. Line Rules Priority Experiments	379
7.4. FOCUS OF ATTENTION RESULTS	383
7.5. SYSTEM PERFORMANCE	389
7.6. CONCLUSIONS AND FUTURE WORK	406
REFERENCES	414
APPENDIX A: THE KNOWLEDGE RULES	424
APPENDIX B: THE CONTROL RULES	436
APPENDIX C: THE STRATEGY RULES	441

LIST OF FIGURES

	Page
1.1. The system block diagram.	7
1.2. Three levels of rules.	11
1.3. Comparison of performance measures for three approaches to segmentation.	21
2.1. The combined use of lines and regions.	39
3.1. Digitized color image of a natural scene.	83
3.2. The Sobel Gradient Operator	84
3.3. Directional masks for the Sobel operator.	85
3.4. Representing gradient direction.	85
3.5. Gradient information for scene in figure (3.1).	86
3.6. An example of a region map.	88
3.7. A partition of the image into regions.	89
3.8. The centroid and minimum rectangle of a region.	91
3.9. Perimeter computation.	93
3.10. Spatial adjacency between regions.	95
3.11. An example of a line partition.	99
3.12. Location and direction of a line.	101
3.13. The spatial relationship IN FRONT.	105
3.14. The spatial relationship BEHIND.	107
3.15. The PARALLEL relationship.	108
3.16. LEFT OF and RIGHT OF relationship between lines and regions.	111

3. 17. The INTERSECT relationship between regions and lines.	112.
3. 18. Example of storing a list of regions for one area.	118
3. 19. Examples of under-merging error computation.	124
3. 20. Two measures of segmentation.	130
4. 1. Initial region partitions.	138
4. 2. Candidate pixels for line grouping.	140
4. 3. A flow chart of the line generation process.	141
4. 4. The merging action.	145
4. 5. Histogram splitting.	148
4. 6. Threshold evaluation.	149
4. 7. Splitting at lines.	151
4. 8. Candidates for line extension.	156
4. 9. Joining two lines.	157
4. 10. Inserting a line at region boundaries.	158
4. 11. Merging Lines.	160
4. 12. The forward merging of two lines.	160
4. 13. The backward merging of two lines.	161
4. 14. Current entries selected by the FOCUS OF ATTENTION MODULE.	166
5. 1. Relative ranges of symbolic qualifiers.	182
5. 2. The structure of a condition.	186
5. 3. The functions of the RULE INTERPRETER.	187
5. 4. Symbolic ranges for features with open-ended scale.	192
5. 5. Symbolic ranges for features with closed scale.	193
5. 6. Symbolic ranges for features with middle reference point.	193

5.7.	Relation between scales for absolute and difference values.	195
5.8.	Computation of lines between regions.	199
5.9.	The Gestalt principles of visual grouping.	210
5.10.	Joining lines using region information.	223
5.11.	Line at the boundary of two textured areas.	225
5.12.	Line at the boundary of two large regions.	226
5.13.	Rules for merging lines.	227
5.14.	The flow chart of a system strategy.	249
6.1.	Example of contrast sensitivity curve.	277
6.2.	Weights as a function of region size.	278
6.3.	Model for strategy evaluation.	301
6.4.	The solution to model F.	311
7.1.	Block diagram of system implementation.	316
7.2.	Actions-versus-cycles curve.	319
7.3.	Rule categories.	320
7.4.	A window in the image of a motor.	321
7.5.	The selected window.	322
7.6.	Region maps for merging rules experiments.	324
7.7.	Merging experiments performance measures.	326
7.8.	Region maps for splitting rules experiments.	330
7.9.	Splitting experiments performance measures.	331
7.10.	Region maps for initial size experiments.	334
7.11.	Total error measures for region size experiments.	335
7.12.	Line maps for deleting lines experiments.	338
7.13.	Performance measures for deleting lines.	340
7.14.	Line maps for extending lines experiments.	343
7.15.	Performance curves for extending lines.	345

7. 16. The effect of regions on line extension.	346
7. 17. Performance curves for the effect of regions on line extension.	347
7. 18. The connected line map superimposed on the initial.	348
7. 19. The effect of using line information on the region maps.	351
7. 20. The effect of lines on region analysis as reflected in the total error measure.	352
7. 21. Test image for strategy experiments.	357
7. 22. First set of region maps for path strategies.	358
7. 23. Region uniformity for region path experiments.	358
7. 24. Second set of region maps for path strategies.	361
7. 25. Region contrast for region path experiments.	361
7. 26. First set of line maps for line path strategies.	363
7. 27. Second set of line maps for line path strategies.	366
7. 28. Connectivity curves for line path experiments.	367
7. 29. Region maps for region updating strategy.	369
7. 30. Error curves for region updating strategy.	370
7. 31. Line maps for line updating strategy.	373
7. 32. Connectivity curves for line updating strategy.	374
7. 33. Region maps for process priority experiments.	375
7. 34. Region uniformity curves for process priority.	376
7. 35. Region maps for region rule priority experiments.	378
7. 36. Region uniformity for region rule priority.	378
7. 37. Line maps for line rule priority.	380
7. 38. Error measures for line rule priority.	382
7. 39. The segmentation result (region map) for the scene in figure (3.1).	385

7.40. The final line partition for the scene in figure (3.1).	385
7.41. The smooth areas detected in the scene.	386
7.42. The textured areas in the scene.	388
7.43. Example of a bounded area.	390
7.44. Partitions of the outdoor scene.	392
7.45. Partitions of the BLOCKS image.	394
7.46. Partitions of the HOUSE image.	397
7.47. Partitions of the GIRL image.	399
7.48. Partitions of the CAR image.	401
7.49. A two-dimensional histogram of performance measures for the three methods.	405

LIST OF TABLES

	Page
3.1. An example of a relation.	81
3.2. Example of the features stored for a region.	96
3.3. Example of line features.	109
3.4. List of lines intersecting a region.	112
5.1. Possible data entries.	185
5.2. Consecutive nearness thresholds.	190
5.3. Region features.	199
5.4. Line features.	200
5.5. Area features.	202
5.6. Other features in a condition.	203
5.7. The rule actions.	204
5.8. Interpretation of processing history.	240
6.1. The dynamic elements of strategy and their states.	264
6.2. Effect of performance parameters on the states of strategy elements.	304
6.3. Codes that reflect the relative importance of performance parameters in determining the various strategy states in different areas.	305
6.4. Symbol table for hedges.	309
7.1. Region merging experiments.	323
7.2. Region splitting experiments.	330
7.3. Initial region sizes and numbers.	333

7.4.	Region size experiments.	333
7.5.	Line deletion experiments.	337
7.6.	Line extension experiments.	343
7.7.	Line merging experiments.	349
7.8.	Effect of lines on regions.	350
7.9.	Rule ranking.	354
7.10.	Region path strategy experiments.	360
7.11.	Relation between performance measures and region path strategy.	362
7.12.	Line path strategy experiments.	364
7.13.	The relation between performance measures and line path strategy.	368
7.14.	Updating strategy for regions experiments.	369
7.15.	Updating strategy for lines experiments.	373
7.16.	The relation between performance measures and the updating strategy.	375
7.17.	Process priority experiments.	375
7.18.	The relation between performance measures and process priority.	377
7.19.	Region rule priority experiments.	379
7.20.	The relation between performance measures and region rule priority.	379
7.21.	Line rule priority experiments.	380
7.22.	The relation between performance measures and line rule priority.	382
7.23.	Comparison of error measures for three methods.	391
7.24.	A comparison of the performance measurements of three approaches.	404

CHAPTER 1

INTRODUCTION

1.1. OBJECTIVES

A primary objective in contemporary computer vision research is the ability to analyse images originating from three-dimensional scenes in order to understand their content. This may be done by assigning appropriate interpretations to objects within the scene. Accordingly, an image must first be segmented into regions that roughly correspond to objects, surfaces or parts of objects in the scene represented by that image. Two stages are thus apparent, the low level processing stage concerned mainly with the segmentation issue, and the high level stage devoted to the interpretation of the segmented result.

The hierarchy suggested here maintains that a low level process that has no a priori knowledge about the objects in a scene, would be able to deliver a "plausible" output to the high level process. However, the option of receiving

feedback from the latter for further processing in order to resolve high level ambiguities is left open. This research is concerned with the low level stage of the image understanding system and is closely related to previous work on high level image interpretation [Levine and Shaheen, 1981]. It should be noted that the distinction between the two levels of analysis is primarily in terms of the knowledge available to each. Whereas the interpretation system uses domain specific knowledge about the contents of a scene, the low level segmentation stage employs general purpose models [Zucker et al., 1975] that contain knowledge about images and grouping criteria that are independent of the class of scenes under analysis.

This thesis proposes a new paradigm for image segmentation, namely, the use of an expert rule-based system. This solution was motivated by a number of issues that needed to be addressed and studied. They can be classified under the following headings:

- (1) The presence of a large number of segmentation heuristics in the literature. A method is required for evaluating and organizing these to work together.
- (2) The absence of an explicit scheme for representing the knowledge embodied by these segmentation heuristics.
- (3) The lack of quantitative evaluations of a segmentation.
- (4) The need for a better understanding of the methods by which the segmentation heuristics should be applied to an image.

Following is a discussion of these issues and how the proposed solution affects each:

(1) Integration of knowledge sources

Approaches to image segmentation in the past have relied on procedural algorithms for boundary detection and region growing [Riesenman and Arbib, 1977]. These processes used different heuristics in their analysis, with varying degrees of success in obtaining output partitions that are compatible with human expectations. The diversity in the data structures, knowledge sources, and control mechanisms presents difficulties in finding a general framework that can integrate these efforts. A primary goal of this research is to design a complex system that not only is flexible enough to accommodate the wide variety of segmentation heuristics, but would also be capable of evaluating and selecting the best heuristics for different data situations. For example, both region analysis for grouping pixels based on the similarity of their features, and boundary analysis for partitioning the image based on discontinuities in the features have been suggested as valid methods for segmenting an image. The proposed system is capable of doing both, with the additional power of allowing interactions between these knowledge sources. During the course of this thesis, we will describe and demonstrate how the exchange of information between various segmentation heuristics leads to a substantial improvement in the output delivered to the high

level interpretation processes.

(2) Explicit knowledge representation

The design of an expert segmentation system that blends knowledge from a wide variety of sources faces the problem of finding a suitable knowledge representation scheme. In vision systems developed so far, "explicit representations of knowledge have been reserved mainly for application domain knowledge, while general visual knowledge has been buried away in procedures" [Tsotsos, 1982]. What is needed is a flexible mechanism capable of experimenting with different processing methodologies, and an expressive coding scheme that can accommodate all possible knowledge sources. The rule-based approach to control which is proposed in this thesis satisfies both requirements. Knowledge is detached from the application processes and coded into the rules, which are modular entities that can be modified without affecting the structure of a system. An additional goal of this research is to investigate the use of the rule-based approach in general purpose image segmentation. This includes addressing control issues such as conflict resolution and focus of attention, as well as structural issues like the symbolic coding of segmentation heuristics into production rules.

(3) Output evaluation

Another aspect of the image segmentation problem is the non-uniqueness of the solution. Ideally, a complete segmentation should correspond to the objects to be interpreted. However, low level processes can only produce partitions on non-semantic basis, and these may not necessarily correspond to describable objects. Such partial segmentations are not unique even for humans examining a scene. This problem may not be of particular importance if the ultimate goal is to perform high level analysis with top-down feedback. Nevertheless, a method for evaluating the output is required. It provides the means for designing a system that produces the most appropriate partition, and hence minimizes the need for intervention by the high level stage. Therefore, another goal of this research is to establish design measures that indicate the distance from a known reference. The latter is taken to be a manually generated segmentation that corresponds to the objects in the image. These measures are then used within the context of the rule-based system to tune the model by evaluating the rules and the control mechanisms.

The development of design measures has prompted us to search for other measures that estimate the quality of the output without the need for a reference partition. Consequently, these new measures can be used during actual processing and will be referred to as "real-time measures". Low level segmentation can now be posed as an optimization

problem in which the goal is to maximize a set of performance parameters. By comparing an output partition to the input image, the real-time measures are used to modify the processing methodology in order to incrementally improve that output.

(4) The definition of a control strategy

For a system that employs a large number of heuristics to segment an image, two control issues must be addressed: the order in which different heuristics are applied, and the path within the image along which they are tested. For the rule-based system, the segmentation heuristics are coded into the rules. A control strategy is defined that consists of rule-ordering and path selection components. Different strategies are implemented and tested. The best strategy to apply varies during processing. It depends on the state of the output segmentation at any given time. Consequently, a dynamic selection mechanism is used to execute the best strategy based on the data. The next section summarizes how this and other issues are incorporated into the rule-based system.

1.2. SYSTEM OVERVIEW

The system is designed in the form of a modular set of processes and two associative memories, as shown in figure (1.1). The input image, the segmentation data, and at the end of processing, the output, are stored in a Short Term

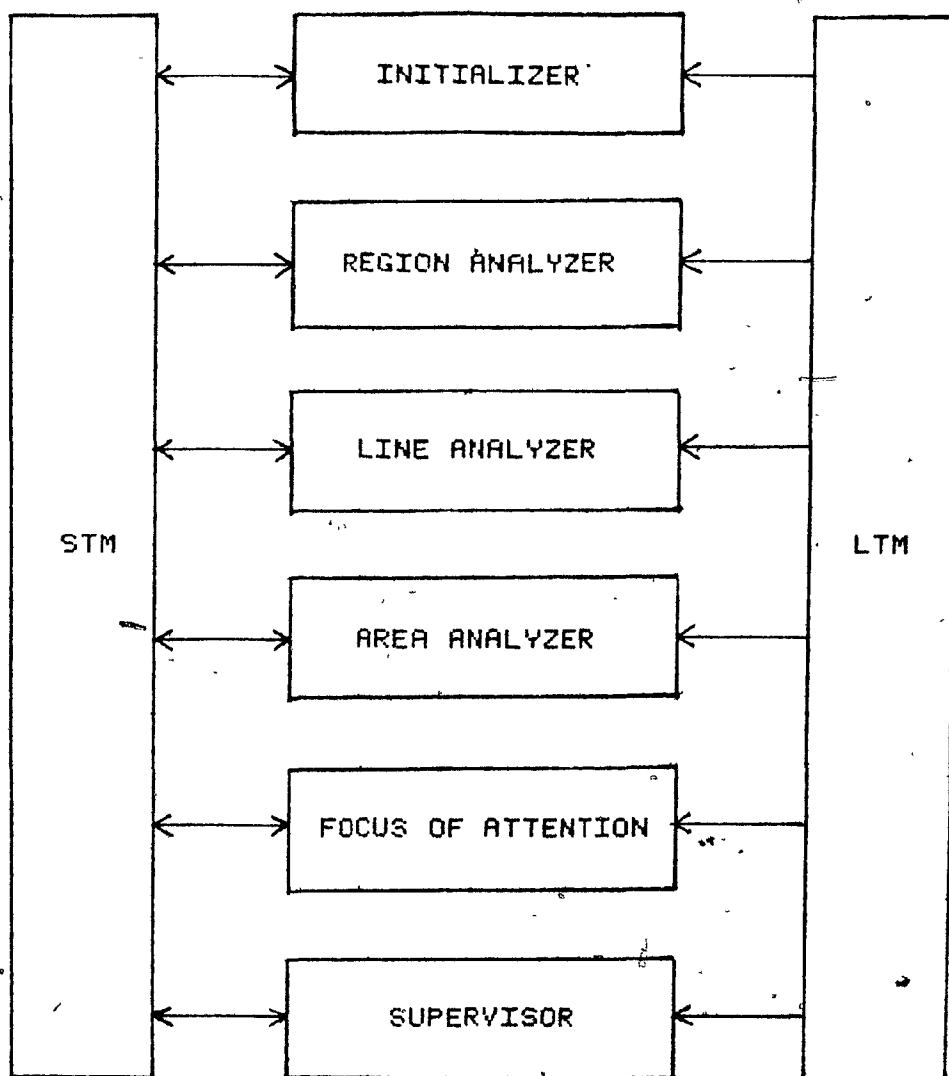


Figure (1.1): The system block diagram.

Memory (STM). A Long Term Memory (LTM) embodies the model representing the system knowledge about low level segmentation, as well as control strategies. Each of the system processes has access to the information stored in the LTM, and can also read and modify the data stored in the STM. In fact, this is the only way it can communicate with the other system processes. Thus any change in the data stored

in the STM, brought about by one of the system processes, will be instantaneously available to the other system processes

The basic control structure is that of a production system, in which general knowledge about independent low level properties of an image is formulated into condition-action rules. A system process matches rules in the LTM against the data stored in the STM, and when a match occurs, the rule fires. This triggers an action that is performed by that process, and usually involves modification of the data.

The segmentation of an image is defined by a set of regions that are connected and nonoverlapping, so that each pixel in the image acquires a unique region label that indicates the region it belongs to. In addition to the regions, discontinuities in the image array are defined by lines that are also connected and nonoverlapping. In this case, pixels in the image can either have a line label indicating the line they belong to, or they can have null labels to indicate that there is no line passing through them. This approach simultaneously uses both region information, based primarily on uniformity criteria in the image array, as well as line information, that represents discontinuity criteria in that array.

Initially the image gradient is computed and thresholded. The resulting edge points are then grouped into lines to produce an initial line map for that image. An initial region map is also computed by either selecting an arbitrary segmentation of the image into regions, or by executing a fast region growing algorithm. The image array, its gradient, and the initial region and line maps are all stored in the STM. Low level features are then extracted and stored in the STM. These include the average color features for regions and lines, position features, and spatial relationships. The latter includes adjacency among regions, and relationships of in front of, behind, and parallel to involving lines. Regions to the left and right of a line, and lines touching or intersecting a region, describe spatial relationships between lines and regions.

Other important data entries maintained and stored in the STM are the focus of attention areas. These are areas in the image defined primarily by their position and size. An area may correspond to a region, a line, or groups of regions or lines in the image segmentation. These areas are used for the purpose of directing the attention of the system to the more interesting and worthwhile parts of the image, in terms of processing need and richness of segmentation information. Three types of areas are defined: smooth, textured, and areas defined by long lines that close, or almost close, forming loops. The latter are referred to as bounded areas. Initially, the process responsible for generating areas,

computes them on the basis of the segmentation data stored in the STM

The information stored in the STM is continuously updated during processing, so that it reflects the state of the segmentation of the image at any point in time. The regions, lines, areas, plus their features at the end of processing, represent the output of the low level stage to be delivered to the high level interpretation system. The higher the correspondence between regions in the output segmentation and objects in the scene, the easier the task of the high level process. Lines corresponding to major discontinuities in the image, as well as smooth and textured areas, can also be used to guide the interpretation process.

The model stored in the LTM is composed of three levels of production rules, as shown in figure (1.2). At the first level, the knowledge rules encode the information about properties of regions, lines, and areas in the form of sets of situation-action pairs. Each situation is coded into a number of logical predicates that compose the conditions of a rule. The latter are either logical comparisons or evaluations on the features of regions, lines and areas in the segmented image. The conditions are ANDed together, so that when a specific situation occurs within the image, all the conditions will be met. In this case, a match is said to have occurred, and the rule fires. The rule action is then executed. This might be a splitting of a region or merging of two regions, an addition, deletion, or extension of a

line, a merging of two lines, or a creation or modification of a focus of attention area. Knowledge rules are classified by their actions, so that there are region analysis rules, line analysis rules, and area analysis rules. A region rule, for example, will contain conditions that match on features of regions, lines, and areas, but will always execute its action on the current region under analysis.

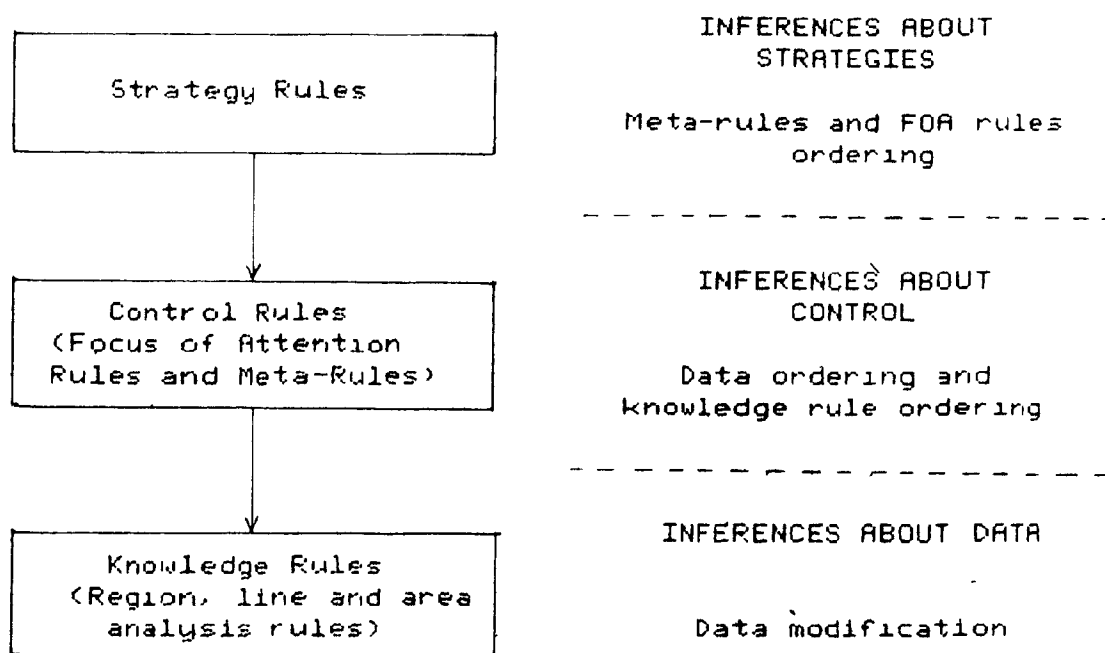


Figure (1.2) Three levels of rules.

The second level of rules in the LTM contains the control rules which can be further classified into two categories. First the focus of attention rules, which are responsible for finding the next data entry to be considered. They have actions that bring to the attention of the system a next region, next line, or an entire area. Using these rules, a strategy for visiting regions and lines within a

focus of attention area, and for alternating from one area to another, can be defined and executed. This strategy will be data-driven, since the conditions of these rules are also logical predicates evaluated on the STM data.

The second type of control rules are actually rules about rules, or in the spirit of a production system, they are termed meta-rules. They differ from other rules, in that their actions do not modify the data in the STM, but rather alter the rule matching order by rearranging the priorities of different rule sets. Since each process in the system is associated with a different set of rules, the effect of the meta-rules is to establish the flow of control between the different processes in the system, by specifying the next process to be activated. The meta-rules are also responsible for evaluating the stopping criteria of the system, and consequently terminating the processing. This approach renders the whole control process to be data-driven. One can effectively "program" the system into executing a desired strategy by putting together the appropriate combination of meta-rules and focus of attention rules.

The third and highest rule level in figure (12) contains the strategy rules. Their function is to select, based on the data, the set of control rules (focus of attention rules and meta-rules) that execute the most appropriate control strategy. The system's control strategy is analysed into six basic elements. The first three are the region path strategy, the line path strategy and the region

and line updating strategy. They determine the order in which the respective data items are tested by the rules. Different path strategies are implemented by employing the appropriate set of focus of attention rules. The next three elements define strategies for determining process priority, region rule priority, and line rule priority. They specify the order in which the different rule sets are matched. In this case, the meta-rules are used to simulate different strategies.

Strategy determination is formulated as a dynamic decision-making process. A set of performance measurements are computed for each focus of attention area in the image. These include values for the average uniformity of the regions within the area, the average contrast across these regions, the connectivity of the lines, and the average contrast across them. These performance measurements are used to determine the appropriate setting for each of the strategy elements described above. For example, if an area has regions of low average uniformity, the region rule priority is set so that rules which split regions are given priority over merging rules. Other constraints define the relation between the performance measurements and all the elements of strategy. The latter will therefore change with time, as well as spatially from one area to another. The strategy rules use this decision-making process to select the set of control rules that accomplish the chosen strategy.

Six processes are involved in producing actions, as shown in figure (1.1). The INITIALIZER generates the initial region and line maps, and uses them to generate initial focus of attention areas. It also computes and stores the features of these initial data elements in the STM. The REGION, LINE and AREA ANALYZERS are the three main processing modules of the system. They are responsible for rule matching and data modification on their respective entities. The LINE ANALYZER, for example, matches the line analysis rules in the LTM to the data in the STM. If a rule fires, it will execute the action specified by that rule on the current line under analysis. The same applies to the REGION and AREA ANALYZERS. The FOCUS OF ATTENTION module matches its own rules to execute a defined path strategy that brings various data items to the attention of the system, in a particular order. Thus, when a focus of attention action is executed, this means that the current region, line, or area will be replaced by a new region, line, or area whose identity depends on the action specified by that rule. Finally, the SUPERVISOR acts as a monitor for system control purposes. It matches the strategy rules to select the appropriate control strategy. Consequently, a set of meta-rules is used to determine which set of knowledge rules, if any, is to be tested next.

The system described above is seen to be both knowledge-based and data-driven. The production system approach to control allows for knowledge separability, and thus provides an easy means for "tuning" the model used for

low level segmentation Rule classification, and the use of focus of attention rules and meta-rules permit easy access to the control mechanisms of the system, without compromising its modularity. A main objective of using such a design approach, is to assist us in understanding the nature and effect of various low level knowledge sources and control strategies.

1.3. ORIGINAL CONTRIBUTIONS

While the details of the research is distributed over the chapters of this dissertation, this section briefly highlights its contributions in the following aspects

(1) Overall system structure

We have designed and implemented a rule-based expert system for the segmentation of images by computer. The basic system structure follows closely that of the high level interpretation system of [Levine and Shaheen, 1981] in that it is composed of a modular set of independent processes that communicate and interact through a common database. What was presented as a single low level process in [Levine and Shaheen, 1981] is expanded to include the set of processes shown in figure (1.1). Whereas the high level model consisted mainly of a set of constraints that drive a relaxation labelling process, the basic control paradigm for the low level segmenter is that of a production system. In fact, the system presented here introduces the rule-based

approach to the image segmentation problem. Previous vision systems limited the use of explicit knowledge to the semantic interpretation level. Consequently, the heuristics used for low level image segmentation were always implemented in procedural form. The system presented here is the first to employ the domain independent knowledge in an explicit form. The knowledge is separated from the processing modules by coding it into production rules and storing them in the LTM. In addition, system control is made more accessible by introducing a set of control rules into the model. This allows a user (or the system supervisor) to interactively (or automatically) program different processing strategies. The separability of knowledge and control represents a unique approach to solving the segmentation problem.

(2) Data structure

Two new aspects are introduced into the data structure. First, regions and lines are combined and are simultaneously used in the analysis. Although other systems have employed both [Hanson and Riseman, 1978], each was processed separately, and the results combined only at the end. In this system, regions and lines share the same data structure and have common spatial features. The model includes rules that manipulate combinations of regions and lines. This allows uniformity and discontinuity information to blend together and cooperate in producing a better output. Experiments with the system prove that this type of

interaction is beneficial

The second aspect is the introduction of focus of attention areas as groupings of the regions and lines that represent interesting parts of an image. In addition to providing valuable information such as texture to a high level process, these areas play an important role in the analysis by providing spatial boundaries across which the processing strategy is adjusted. This allows the system to execute the best strategy based on the individual properties of each area.

(3) The rule-based model

A multitude of segmentation criteria are coded into the rules. These include the individual heuristics that formed the basis of earlier segmentation techniques. In addition, other general purpose knowledge sources are employed, such as the Gestalt principles for perceptual grouping.

The system also introduces a general symbolic coding scheme for the rules. The conditions are logical predicates evaluated on the STM data by using an automatic scaling mechanism. Low level symbolic processing is a step towards reducing the dependency of previous segmentation methods on numerical parameters. Thus, model learning through rule modification replaces the more ad hoc process of interactive parameter tuning.

The actions of the rules are independent entities that are carried out by the processing modules. They are designed to modify the regions, lines, and areas in finite ways, yet they have the flexibility to permit changes in different directions (e.g., merging and splitting of regions).

The conditions and actions create a "segmentation language" that provides for a wide range of data manipulation and modification. The ability to modify and expand the knowledge base is an important contribution that is a result of the symbolic coding scheme and the rule-based structure. The testing and tuning of the model presented in chapter 7 was only possible because of this powerful experimental feature. Furthermore, future addition and testing of new segmentation heuristics require minimum user interaction.

(4) Measures of segmentation

A primary concern of this research is the ability to evaluate intermediate and final output partitions. A set of error measures was devised that dynamically computes the distance between a test partition and a reference segmentation. These are used to build the model by testing different rule combinations and selecting those that are the most effective in reducing the errors.

A second set of measures are unlike any developed so far in that they do not require a reference segmentation. Performance is judged by comparing the segmentation to the

original image. This enables the system to have "real-time" estimates of the incremental development in processing. These performance measures behaved consistently with the segmentation error measures throughout a wide variety of experiments with the system.

(5) The control mechanism

A strategy determination process that varies with time and with image area has been introduced. The elements of control strategy were defined and their dependence on the real-time performance measures was experimentally verified. The experiments tested the effect of using different control strategies on the performance of the system, in terms of output quality and efficiency of computation. Analysis of the results lead to conclusions about the best strategies to apply for different ranges of performance measures. Such constraints are used in executing the dynamic strategy selection process.

This dynamic strategy mechanism presents solutions to two control issues. First, for rule-based processing, conflict resolution is achieved by creating a dynamic data-driven rule re-ordering process. Second, a focus of attention mechanism is created that is capable of adjusting its path of processing within the image according to the data. These factors result in a system that can select the most suitable segmentation heuristics and apply them to the parts of the image where they are most needed.

(6) Experimental results

The ability to quantitatively measure performance allowed us to conduct experiments with various knowledge rule combinations. A ranking of the rules was then established based on their effectiveness in maximizing the performance measures and their frequency of firing. The contribution of these experiments is the selection of an optimal set of rules over a large collection of segmentation heuristics.

The performance of the system operating with dynamic strategy setting and the optimal set of rules was compared to other segmentation methods in the literature. The results show a considerable improvement in the quality of the output, in the segmentation error measures, and in the performance measures. The graph in figure (1.3) summarizes these results. It indicates the range of performance parameters measured on the final output partitions of several images for three different segmentation approaches. Algorithms that employ grouping based on similarity result in partitions that are highly uniform. Histogram-based partitions exhibit high contrast between the regions. Each sacrifices one measure to enhance the other. On the other hand, rule-based segmentation produces partitions that simultaneously maximize uniformity and contrast. The details of these results will be presented in chapter 7.

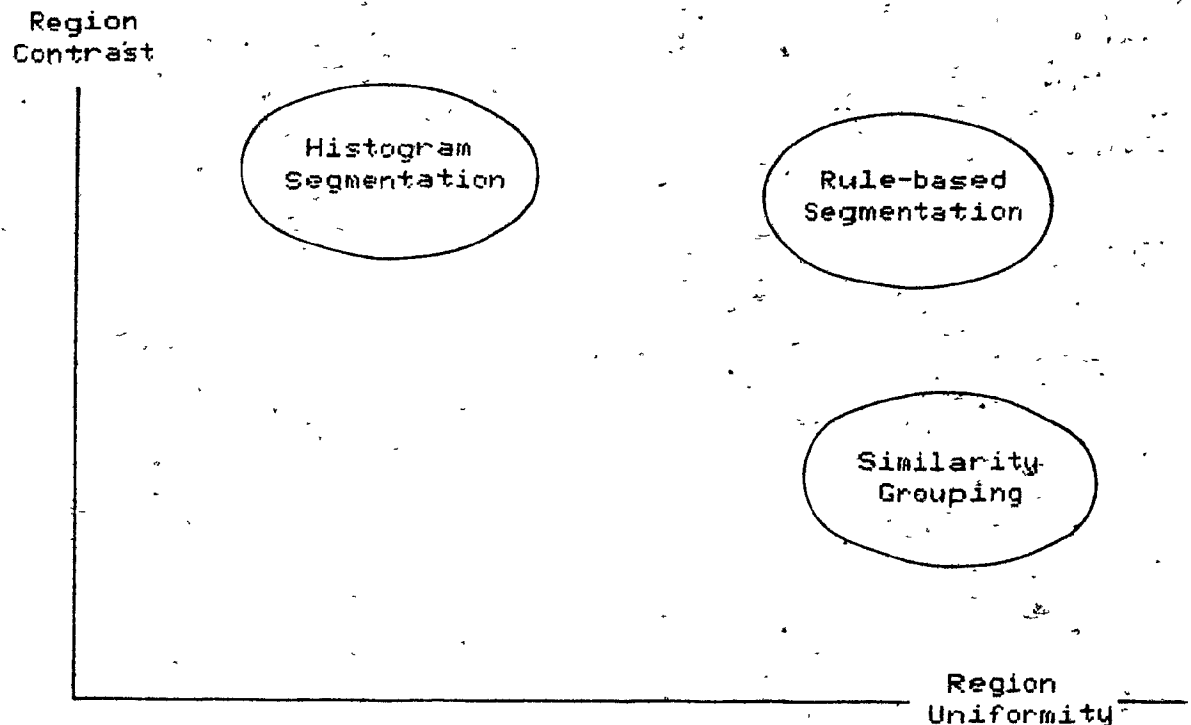


Figure (1.3) Comparison of performance measures for three approaches to segmentation.

1.4. THESIS DESCRIPTION

Chapter 2 reviews previous efforts in the field. A selective survey of research work in the areas of computer vision systems and knowledge-based representations is conducted. Image segmentation approaches that use region analysis and boundary detection are reviewed, and the hierarchical structure that has been established for computer vision is examined. This is followed by a brief description of the basic properties of production systems. In particular, how the latter evolved into the more complex rule-based systems that have been designed for various

artificial intelligence applications. The few attempts at employing rule-based systems for computer vision are then evaluated. Finally, the suitability and usefulness of the rule-based approach for general purpose image segmentation are studied.

The data structure is introduced in chapter 3. Image specific information stored in the STM is divided into three categories: the static image data, the segmentation data, and system information and statistics. The region and line partitions are defined together with the descriptive features and spatial relationships that are computed and stored for each entry. The different types of focus of attention areas are then described, and the method by which they access groups of regions and lines is explained. A set of measures is presented that quantitatively evaluates segmentation outputs by comparing region and line partitions to human generated references. These error measures are used in chapter 7 to test and tune the knowledge base, as well as to compare between different image segmentation approaches.

In chapter 4, the processing methodology is described in terms of the six modules that constitute the rule-based segmenter. Functional descriptions are given of the INITIALIZER, the REGION, LINE, and AREA ANALYZERS, the FOCUS OF ATTENTION module, and the SUPERVISOR. The actions that merge and split regions, add, extend, and merge lines, generate and maintain focus of attention areas, and perform the focus of attention and rule selection strategies are all

described in detail. Every module has a set of actions associated with it, which are invoked by the LTM rules to execute specific procedures that affect the data in the STM.

The structure and content of the LTM rules are explained in chapter 5. A scheme for the symbolic coding and evaluation of the logical predicates that form a rule condition is presented. The different types of knowledge rules are surveyed and the detailed criteria behind each rule are examined. The focus of attention rules and the meta-rules are then discussed, and examples are given of how they are used to program different control strategies.

Chapter 6 defines the elements of the control strategy for the rule-based system, and associates specific states with each. A set of performance measurements that is independent of a reference segmentation is then introduced. This is followed by a brief introduction to the principles of fuzzy logic. The constraints that govern the relation between strategy elements and performance measurements are defined as fuzzy variables. A dynamic decision-making process is then described that uses these constraints for strategy determination.

System implementation and experimentation are the focus of the analysis in chapter 7. The performance measures, the segmentation error measures, and the output partitions are used to compare and rank the knowledge rules. The constraints that govern the selection of the control strategy

are derived from the performance patterns in experiments with all strategy elements. Results with different types of focus of attention areas are then presented. Overall system performance is judged in comparisons with existing segmentation techniques. The thesis concludes with a brief assessment of the results and suggestions for future research.

CHAPTER 2

COMPUTER VISION AND RULE-BASED SYSTEMS

2.1. BACKGROUND

One of the goals of this research was to investigate the use of rule-based systems in computer vision. Consequently, two domains of study are involved, both of which belong to the field of artificial intelligence. First, there is the computer vision problem, an application oriented AI domain that has well-defined goals but no clear means of achieving them. Efforts in the field have been substantial, yet a unified theory for computer vision has not been established. On the positive side, early attempts provided insights into the complexity of the problem, and proved the impotency of solutions that are too simplistic or those that work for very restricted inputs. More recently, a consensus has been reached that in order to simulate the visual task, more sophisticated issues have to be addressed [Zucker, 1981, Marr, 1982, Levine, 1983].

The second related AI domain is that of rule-based systems, a mechanism that has been successfully deployed in other AI applications. What started as a general computational tool known as production systems, became a well developed field on its own when researchers used them to build large application-oriented programs. Modifications and variations in the basic production systems design were inevitable in order to accommodate the complicated data structures and achieve the multiple goals set by these applications.

The system presented in this thesis uses this popular AI mechanism as its basic computational device. The rule-based approach has affected the structuring of this vision system both in terms of control and data representation. This chapter provides the necessary background that has led us to bring both domains together. First, we present a brief review of the computer vision field and explain the need for the framework provided by the rule-based approach. We then study the characteristics of rule-based systems while examining programs in other application areas. Few efforts have preceded this one in using production systems for image processing applications. These will be studied in detail, pointing out their limitations and the need for a more general approach. Finally, the suitability of production systems for computer vision is examined, and the advantages gained by using this approach are listed.

2.2. COMPUTER VISION SYSTEMS

Image analysis by computers has become an active field of research over the last two decades. In the fifties, research was initiated into the recognition of simple structures in two-dimensional scenes. A major area of application then and now is that of character recognition (printed and handwritten). The need for enabling machines to identify objects in images of more complex scenes stemmed from applications in industrial automation, biomedical engineering, remote sensing, and robotics. Research in image understanding has since produced a large number of theories, approaches, and proposed solutions. Yet, the results obtained are in disproportion to the efforts invested. The realization that a single-shot solution that is based on a few heuristics is not the answer has recently become apparent. The trend is now to design large scale systems that are capable of encompassing a wide variety of knowledge sources and that produce partial (intermediate) solutions that are in the right direction. The work described in this thesis is an integral part of this approach.

In this section, we present a brief review of the major highlights of research in image understanding. Although this is not intended as an exhaustive survey, it is hoped that it will provide the necessary background for the work presented here and will show the major trends that led efforts in the field up to its present state. Rosenfeld has consistently provided extensive yearly reviews of the published material

on image processing by computers [Rosenfeld, 1977, 1978, 1979, 1980, 1982]

2.2.1. Line Analysis

The work of Roberts marks the beginning of research in building general vision systems [Roberts, 1965]. Even at this early stage, he proposed a complete framework of an image understanding system that had at its input a digitized image, and delivered a three dimensional description of objects in the scene represented by that image. The domain of the system was restricted to the world of planar surfaced, simply connected polyhedra, better known now as the "blocks world". Simple heuristics were used to produce a line drawing from the digitized image. These included edge detection followed by the grouping of edge points into lines. Failing to produce an adequate line drawing, Roberts assumed the presence of a perfect output to be utilized in the subsequent stages of his program. The higher levels of processing compared the line drawing to a set of models of stored polyhedra, recognizing objects sequentially and deleting them from the scene.

This initial effort is significant in more than one way. First, it proposed a hierarchy of processing that starts at the input image, and proceeds through higher levels of abstraction, reaching the symbolic description of scene content at the top. In spite of the large number of efforts in the field, little change has affected this organization.

which still constitutes the basic backbone of all systems presented to date. Second, it illustrated the presence of a gap between the preliminary processes of digitization and edge detection on the one hand, and the high level semantic-based recognition processes on the other. This gap is due to the inadequacy of simple edge detection and line grouping algorithms in providing a suitable output that can be successfully manipulated by the high level processes. Finally, the restricted domain of analysis became the subject of many of the efforts that followed. Blocks world programs have given researchers a lot of insight into the complexities of the problem at hand, although some argue that it hindered progress in the field by concentrating efforts on difficult, but less important problems that are products of this artificial world [Tenenbaum, 1973].

Other blocks world programs soon followed. Guzman used simple heuristics to interpret perfect line drawings of relatively more complex scenes [Guzman, 1968]. Huffman and Clowes studied the types of vertices and edges that occur in the blocks world as well as the legal combinations that form physical surfaces [Huffman, 1969, Clowes, 1971]. They introduced a labelling scheme that constrained the number of possible interpretations of a vertex. By propagating the analysis from the vertices to the surfaces, it was possible to determine whether an object is physically realizable in three dimensions. Falk's program interpreted noisy line drawings of polyhedra based on nine fixed-size object

prototypes [Falk, 1970]. Shirai analysed line drawings of brick configurations using heuristics that predicted additional lines in partially completed drawings [Shirai, 1975a]. Waltz devised a filtering algorithm that utilized a vertex labelling scheme similar to Huffman's and Clowes' to assign consistent labels to the lines and vertices in a line drawing [Waltz, 1975].

As with Roberts' program, other blocks world programs either used perfect (simulated) line drawings as their input, or fell victim to the imperfections, noise, and incompleteness of the output of simple low level processors. For this reason, edge detection and boundary formation have been the subject of extensive research (see [Davis, 1975, Riesenman and Arbib, 1977; Nazif, 1978] for reviews of the literature).

Most approaches attempt to compute the gradient (first derivative) of intensity in the image array. Spatial differentiation is a local operation that is accomplished through the use of two dimensional masks that are convolved with the image. Examples of these include the Roberts Cross Operator [Roberts, 1965], the Prewitt Operator [Prewitt, 1970], the Kirsch Operator [Kirsch, 1971], and the Sobel Operator [Duda and Hart, 1973]. The convolution of these masks with the image produces a gradient image. Pixels with low gradient values indicate regions of uniform intensity in the image, whereas high values correspond to changes in intensity levels at the boundaries between regions.

An edge image is then computed by applying a threshold to the gradient image, so that a pixel represents an edge point if it corresponds to a higher value of gradient, and vice versa. The choice of the threshold is the major stumbling block with these methods of boundary detection. A single absolute threshold value will almost always fail in detecting all relevant edges in an image without picking up artifactual edges and other noisy edges. Adaptive thresholds have been used as a function of the average gradient in a window of size $n \times n$ centered at each pixel [Haralick and Shanmugan, 1973]. Experiments with both absolute and adaptive thresholding have shown that the resulting edges are extremely sensitive to the threshold value and the window size [Nazif, 1978; Levine, 1980].

Spatial differentiation produces multiple indications of boundaries, because the edges do not always correspond to step changes in intensity. The nature of the digitized data, as well as the effect of illumination and shadowing result in incomplete, as well as redundant representations of edges. Operators have been introduced for the suppression of redundant edges by deleting multiple edge points with non-maximum gradient [Rosenfeld, 1969]. Other approaches attempt to group edge points into lines and connect incomplete lines by inserting edge points in the gaps between detected boundaries [O'Gorman and Clowes, 1973; McKee and Aggarwal, 1975; Ramer, 1975; Shirai, 1975b; Perkins, 1976; Baird, 1977].

The preceding edge enhancement concepts have been generalized and embodied in a process of competition and cooperation known as continuous relaxation labelling [Rosenfeld et al, 1976, Zucker 1977, Zucker et al, 1977]. A label is assigned to each pixel to define its likelihood of being an edge point in each orientation. The process then iterates in updating the likelihoods of all labels at each pixel based on those of its adjacent pixels within a specified neighborhood. Likelihoods are increased or decreased according to whether continuity and thinness constraints confirm or suppress the assertion of an edge at the given orientation. Respectively Relaxation procedures are capable of connecting line segments and suppressing multiple edge indications. Their application to images of natural scenes show a considerable improvement over the output of edge detectors [Riseman and Arbib, 1977]. The quality of the results depends on the form and extent of the neighborhood functions that define adjacency. If these are extremely local, the algorithm becomes more sensitive to noise and less capable of resolving ambiguities. On the other hand, a large neighborhood allows for more global information to propagate through the image at the expense of increasing the computational complexity.

Instead of detecting edges by locating points with maximum first derivative, Marr and Hildreth proposed a method that depends on finding the zeroes in the second derivative of intensity [Marr and Hildreth, 1980]. A blurring operation

was applied by convolving the image with a Gaussian filter. The second derivative was then computed using a Laplacian operator, and the zero crossings of the resulting image were detected. The cascaded operation can be approximated by a difference of Gaussians (DOG) function, which has a two-dimensional profile similar to a Mexican hat.

What made this approach attractive was its apparent resemblance to the mechanisms of edge detection in humans [Wilson and Bergen, 1979]. Because of the nature of the zero crossings, the resulting edges are forced to form closed curves, a property that is not always true for physical edges. A second disadvantage lies in the dependence on the mask size (the width of the smoothing filter). Larger masks will blur details and detect major boundaries, while smaller masks will pick up more detailed edges. The problem is that more blurring may cause large boundaries that are spatially close to merge and distort one another. On the other hand, less blurring makes the operator more sensitive to noise. Moreover, a single mask size is clearly not sufficient. Therefore, Marr and Hildreth suggested the use of a multi-level operator that combines the outputs from several mask convolutions to produce a meaningful edge image.

A similar problem exists when detecting texture edges [Rosenfeld and Thurston, 1971, Rosenfeld et al., 1972]. A window size must be selected in order to sample the texture features (e.g., edge per unit area) within each region. If the chosen windows are not large with respect to texture

elements. the measure obtained will not be invariant across the textured region. On the other hand, if the windows are large, they cannot always be expected to fall within a region. This leads to the extraction of hybrid values due to the overlap of two or more regions. Approaches to overcome this problem suggest that window size selection should be a dynamic process [Rosenfeld and Thurston, 1971, Uhr, 1972, Marr, 1976, Tanimoto and Pavlidis, 1975, Levine and Leemet, 1976]. In chapter 6, we will present a solution that eliminates the need for fixed size windows when computing texture measures.

The preceding approaches used edge detection as a means for segmenting an image. They depend on the analysis of discontinuities in the intensity array to produce a line drawing of the boundaries of objects in the scene represented by that image. In the next section, we will focus on the complementary approaches of finding regions that correspond to object surfaces.

2.2.2. Region Analysis

This type of analysis relies on grouping together pixels that constitute areas of uniform feature values. Surveys of region-based segmentation techniques can be found in [Zucker, 1976; Riseman and Arbib, 1977, Kanade, 1978].

One of the earliest approaches grew regions by merging nuclear areas (2x2, 4x4, or 8x8 pixels) based on the similarity in their statistical features [Muerle and Allen, 1968]. The process proceeded in a sequential scanning pattern starting at the top left corner of the image and used grey level intensity as the sole feature for computing similarity.

Brice and Fennema used a similar process to generate initial regions consisting of connected pixels of equal intensity [Brice and Fennema, 1970]. They defined two heuristics for merging atomic regions based on the length and the strength of the boundary that separates them. The resulting regions were strongly dependent on the thresholds required for applying the merging heuristics.

Yakimovsky and Feldman used a decision theoretic approach that is based on introducing semantic information to the region growing process [Yakimovsky and Feldman, 1973]. They computed a probabilistic estimate of the object interpretations of each region according to its features. The probability that a boundary separates two regions was then computed based on the properties of the boundary and the two regions defining it. The merging process successively collapsed boundaries having probability values that were below a given threshold. This system demonstrated both the usefulness and the difficulty of using semantic knowledge in the segmentation process.

Another approach that used semantic information to aid region growing was proposed by Tenenbaum and Barrow [Tenenbaum and Barrow, 1977]. A merging algorithm similar to Brice's and Fennema's was employed to grow the regions. Identity labels were interactively supplied to initial regions and were inherited when a region was merged with another that was not labelled. A merger of two regions with different labels could be manually blocked, while a label was provided by the user for regions that grew above a certain size threshold. The interpretation process utilized a constraint satisfaction mechanism similar to Waltz's filtering algorithm. Riseman and Arbib pointed out the difficulties of automating this approach [Riseman and Arbib, 1977].

A different approach to region analysis is based on successively partitioning the image into increasingly uniform areas. This is accomplished by thresholding the histograms of features in the image, usually intensity, color, and texture. Ohlander developed a technique for choosing the best peak in a set of one-dimensional histograms of various features [Ohlander, 1975]. He defined a set of heuristics for peak selection that set thresholds at the valleys on both sides of a peak. The regions whose feature value fall in the range between the thresholds were then given the same symbolic label, and the process iterated. In order to achieve plausible results, textured regions had to be excluded from the analysis. The major flaw with histogram

splitting techniques lies in the loss of spatial information in the histogram computation. In addition, two distinct regions can contribute peaks that overlap in such a way so that the resulting histogram is unimodal. Other approaches to histogram analysis used less ad-hoc methods for threshold selection [Nakagawa and Rosenfeld, 1979]. Hanson et al. described a method that employed cluster analysis in two dimensional histograms, and preserved spatial relations by computing histograms over fixed size windows [Hanson et al., 1975]. This inevitably lead to the window size problem discussed in the previous section. Consequently, an attempt was made to suppress non-maximum feature values in order to achieve better cluster separation.

A technique that combined both merging and splitting of regions was first described by Horowitz and Pavlidis [Horowitz and Pavlidis, 1974]. The algorithm divided the image into $n \times n$ non-overlapping squares, where n varied from one pixel up to the size of the image. This defined a pyramid data structure [Riseman and Hanson, 1974, Tanimoto and Pavlidis, 1975], with every node at a certain level (resolution) having four sons at the next lower level (higher resolution). Starting with an arbitrary configuration of squares, the algorithm merged those with similar properties that are adjacent and have the same father node at the higher level. This was followed by a process of splitting squares into their four sons if they had large statistical error norms. Merging and splitting were performed separately and always in

that order. A grouping operation was performed on the final output to merge similar regions that had different ancestors but were spatially adjacent. This final merging operation required a spatial search of the image, and in effect, reduced the algorithm into one that merged regions based on their similarity. The simultaneous implementation of the two region analysis techniques was a step towards a more flexible solution to the segmentation problem. However, the restrictive data structure and the fixed order of processing were factors that limited performance in order to achieve computational simplicity. The rule-based segmentation system described here employs both splitting and merging within a much more flexible data structure and in a virtually unrestricted temporal order.

Whereas the approaches discussed so far used either lines or regions in the analysis, Hanson and Riseman described a low level processing stage for their image understanding system that employed both [Hanson and Riseman, 1978]. A Kirsch operator [Kirsch, 1971] was used to generate an edge picture. This was followed by a relaxation labelling process [Zucker et al., 1977] that grouped edges into boundaries. Regions were formed by analyzing clusters in the histogram of color features using a parallel iterative relaxation procedure to assign each pixel to one cluster (the cluster centers were manually specified). The resulting regions and lines were then integrated and used as input to their VISIONS semantic analyser. Lines and region boundaries

were combined if they fell within an acceptable disparity range. The latter was determined by the spatial extent of the gradient surrounding the boundary. This system acknowledged the importance of using both line and region information in the analysis, mainly because they are based on different criteria (i.e., discontinuity versus uniformity). However, each is computed separately, and their combined use is postponed until the later stages of the analysis, as shown in figure (2.13). The system proposed in this thesis provides a new and more powerful approach that integrates region and line analysis at the lowest level. Processing of regions and lines proceed simultaneously with region information used in line analysis, and vice versa, as shown in figure (2.1b).

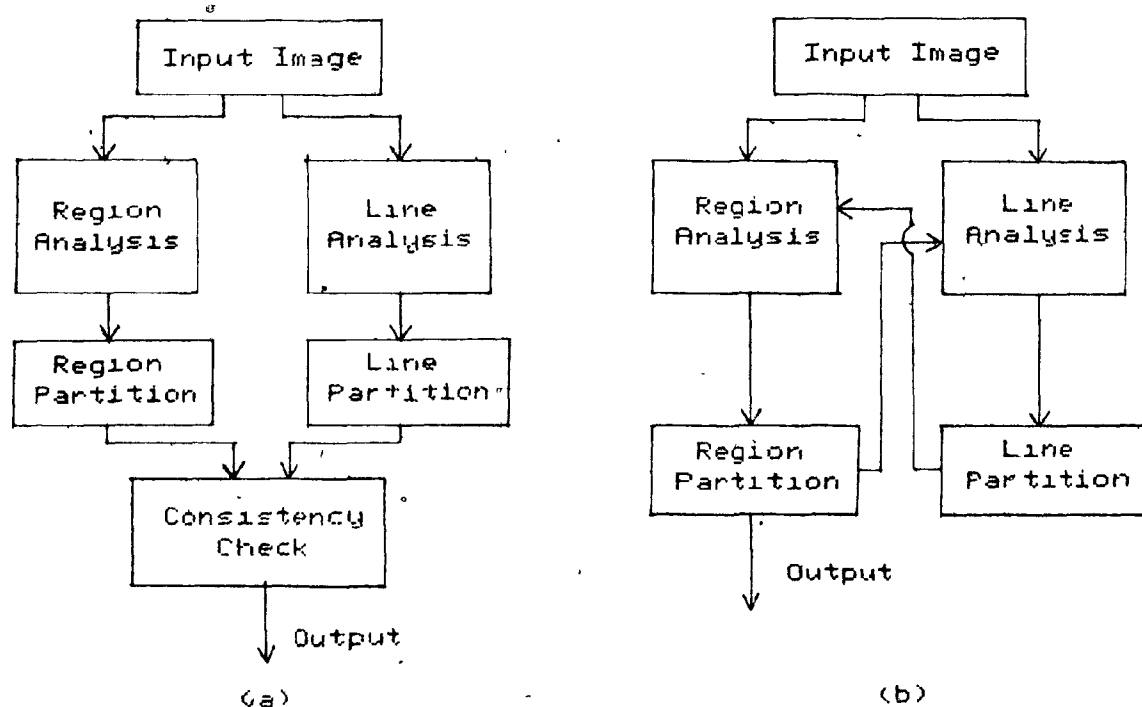


Figure (2.1) The combined use of lines and regions
 (a) Integration at the output
 (b) Integration during processing.

A wide variety of results have been reported using the approaches described above. Although some subjective judgement can be rendered by examining the output of these segmentation algorithms, there is a need for a quantitative method of output evaluation. In chapter 3, we will introduce a set of measures that compute the difference between two image partitions. These will be used to estimate the errors in various segmentation outputs by comparing them to reference outputs generated by hand. Common errors are due to uniform regions that are over-segmented, other regions that require further partitioning, missing lines and detected lines that do not correspond to actual boundaries. The following shortcomings contribute directly to this less than ideal outcome.

(1) Knowledge Sources

The use of a limited number of heuristics that are incompatible with the complexity of the problem at hand. A brief review of the efforts will quickly show that a large number of segmentation criteria has been developed for segmenting images. Yet, each approach contributed and utilized only a fraction of this knowledge. Our efforts were motivated by a desire to integrate the wide variety of knowledge sources into a single solution through the use of the rule-based approach.

(2) Search Order

Most techniques ignore the importance of the order in which the pixels in the image should be processed. This is especially true of region growing algorithms, where the order in which regions are merged affects the final outcome. Zucker has presented the graph theoretic conditions under which the above statement holds [Zucker, 1976]. Order independence can only hold if the sequential region growing algorithm is equivalent to a parallel algorithm that decides on all its merging actions before performing any of them. Experiments have shown that changing the order in which the data was presented to the same set of segmentation criteria produced different results [Nazif and Levine, 1981]. In this research, we employ a focus of attention mechanism that executes a data selection strategy for choosing the regions and lines to process. Furthermore, this strategy specifies the path of processing through the image and will be shown to be dynamic and data driven. The focus of attention mechanism will be described in chapter 5, and dynamic strategy setting will be dealt with in chapter 6.

(3) Control Structure

Systems were designed in the form of procedures that were applied sequentially to the image with little room for feedback between the various stages of analysis. A general purpose low level stage should be able to interact with higher levels that use domain specific information in order

6

to refine the image partition fed forward to these semantic processes. The next section deals extensively with this point.

2.2.3. The Vision Hierarchy

Ever since the work of Roberts [Roberts, 1965], computer vision systems have been structured in a hierarchy of stages representing the transformation from the raw image data to the symbolic description that constitutes the perceived output. In general, most systems distinguish between two main levels of analysis. A low level deals directly with the measured data (the input image) and produces a more concise description of the relevant information. This output is then used by a second and higher level whose processes attempt to describe the image content in terms of a world model. Region growing and edge detection are examples of low level processes that reduce the amount of data to be handled by higher levels. Both create a partition of the image whose description replaces the pixel information present at the input. A high level process can use the partitioned image to assign object interpretations to each of the regions [Levine and Shaheen, 1981]. In this sense, the vision problem can be viewed as a data reduction process.

Although the above hierarchy applies to almost all vision systems designed to date, they differ in the amount of interaction and the extent of overlap between the levels. Yakimovsky and Feldman's system represents the view that

supports the early use of semantic information in the recognition process [Yakimovsky and Feldman, 1973]. An a priori model must be constructed of the objects that are present in the class of scenes under analysis (e.g., outdoor scenes). A large number of models is required to cover the wide variety of possible scene classes. For general purpose vision, a search must be conducted for a specific model for each image. The larger the amount of data to be dealt with, the more complicated this search problem becomes.

Modular approaches separate the processing levels in a more distinct fashion. The low level stage uses information sources that are independent of the type of scene to be analyzed. These include knowledge about intensity changes in the image, possible grouping criteria, noise, and artifacts that are due to image formation procedures. They constitute what has been termed general purpose models [Zucker et al., 1975]. The low level processes reduce the amount of data to be handled to the extent that allows a search for the appropriate semantic model. The expert high level system then uses the selected model to interpret the low level output.

Because of the failure of early efforts in producing a plausible output for simple domain tasks, it soon became apparent that a simple sequential processing methodology was not sufficient. This suggested the use of feedback of information from higher levels within the image understanding scheme to provide global guidance to the more local low level

processes [Uhr, 1972, Marr, 1976, Levine, 1978, Hanson and Riseman, 1978]. Feedback paths between different levels of analysis can be provided by unifying the data structure and the method in which knowledge is represented [Nazif, 1978]. The rule-based approach described in this thesis provides a suitable medium for the interaction between levels. Each process has its own rules, and the communication channel is provided by the data. These concepts will be analysed in more detail in the next section.

Marr argued that a free-standing low level system that does not depend on a specific semantic model is in fact realisable [Marr, 1976]. His reasoning about the failure of previous systems was two fold. First, the amount of information used was not sufficient, since changes in intensity are not the only factors of interest in the input image. Second, a more flexible representation is required in order to accommodate all possible nuances in an image. The richer the vocabulary used for image description at all levels, the less will be the need for making uncertain assertions to interpret the given information.

The representational framework for deriving shape information from images proposed by Marr [Marr, 1982] is a typical example of the hierarchical approach adopted by many researchers in the field. The framework existed at four levels of abstraction: first, the image with intensity values at each point, second, a "primal sketch" that contained explicit two dimensional information about

intensity changes in terms of zero-crossings, blobs, edge segments, virtual lines, and boundaries, third, a 2 and 1/2 D sketch that described orientation and depth of surfaces in a viewer-centered coordinate system, and finally a 3-D representation of the shapes and their spatial organization in an object centered coordinate frame (e.g. a stick-figure representation).

Since we are concerned with the early stages of visual processing, our data structure exists at the level of Marr's primal sketch. However, the segmentation data that will be described in chapter 3 differ in some aspects from the constituents of the primal sketch. In addition to discontinuity, uniformity of features is considered an important source of information for image segmentation. This is represented by the inclusion of lines and regions in the data structure. The basic principle behind creating a large and complex data structure is closely related to the idea that "at the early stages of the analysis, the representation used depends more on what is possible to compute from an image than on what is ultimately desirable" [Marr, 1978].

One of the intermediate stages that is gaining attention today is the recovery of three-dimensional information from the image. The underlying concept is that image intensities are a function of the illumination of the scene, as well as the characteristics and orientations of the surfaces from which light is reflected [Horn, 1977]. By making appropriate assumptions about the light sources, the surface reflectance

properties, and the geometry of the objects, surface orientation can be computed from image intensities. This task has proven quite difficult even in a simple scene domain, as is evident in the Retinex theory which attempts to compute surface reflectance for Mondrian pictures [Land, 1977]. A key issue is the study of the constraints required to recover the information from the image [Zucker, 1981, Levine 1983]. For example, uniqueness and continuity have been suggested as reasonable assumptions about gradients of surface orientation [Woodham, 1977].

Barrow and Tenenbaum described regions and edges in the scene in terms of the characteristics of the surface elements that are visible at each point in the image [Barrow and Tenenbaum, 1978]. Reflectance, orientation, incident illumination, and range are all "intrinsic characteristics" that are useful for higher level scene analysis. Kanade has termed this intermediate stage of assigning physical attributes to the low level image partitions "the physical level of processing" [Kanade, 1978].

A similar approach that employed semantic information in computing surface orientation has been proposed [Douglas, 1981]. Object labels were initially assigned to the regions produced by the low level process. Depth cues were then used to establish orientation estimates for the regions. An iterative algorithm grouped regions into surfaces, and a semantic net of object models assigned surfaces to specific objects. This type of 3-D region interpretation constitutes

a useful addition to an image understanding system

Research in image understanding is being conducted at the Computer Vision and Graphics Laboratory of McGill University [Levine and Leemet, 1976, Levine, 1978, Levine and Ting, 1981, Levine and Shaheen, 1981, Levine and Nazif, 1982a]. The ultimate goal is to build a general image understanding system that is highly modular, and has large experimental capabilities. These features allow vision researchers to experiment with various knowledge sources and control strategies in order to determine those best suited for the vision task. The hierarchy is well defined and starts with a low level segmentation system [Levine and Nazif, 1982a] that is followed by a high level interpretation system [Levine and Shaheen, 1981]. Intermediate stages have also been studied [Levine and Ting, 1981]. The research reported in this thesis is an integral part of this effort and deals with the low level stage of the analysis. The system is designed to use segmentation rules, and provides knowledge separability, flexible control, and high experimental capabilities.

The high level interpretation stage discussed in [Levine and Shaheen, 1981] was designed as a process of competition and cooperation that iteratively assigns object interpretations to the regions obtained from the segmentation process. A specific world model of the objects in the scene is used to guide a relaxation labelling process into finding the best set of interpretations for the given partition. The

results obtained show a sensitivity to the quality of the initial low level image partition. This has motivated the development of a more sophisticated segmentation system, not only to deliver a better output, but also to open the way for a more comprehensive interaction between both levels of analysis. The reader is referred to [Shaheen, 1979] for a detailed look at the high level interpretation system.

The vision hierarchy adopted here is the one that has been maintained by most researchers since the early days of computer vision. A representation that goes from pixels to regions and lines, and ultimately to a symbolic description of the objects in the scene. In the next section, we will investigate the rule-based approach in AI and show its usefulness in solving the low level segmentation problem.

2.3. KNOWLEDGE-BASED SYSTEMS

We start our discussion by indicating what the term "knowledge-based" implies. The terminology is somewhat vague, since all systems incorporate some form of knowledge and are, therefore, knowledge-based. It is generally accepted that the knowledge used has to be well defined and easily identifiable within the basic structure of a system. One way of achieving this is by separating the knowledge from the control mechanism. Rule-based systems are a subset of knowledge-based systems that successfully meet the above criterion. The rules embody the knowledge and constitute the world model. Production systems (PS) form the building block

of all rule-based systems existing todate. They were first introduced by Post [Post, 1943] as a general computational mechanism that has the capability of a Turing machine. Since then, a lot of deviation from the basic PS characteristics has been introduced in order to accommodate the large AI programs that employ this methodology.

In this section, we provide a brief description of the fundamental PS properties. This is followed by a short review of some of the important expert systems, and the aspects in which they differ from the so-called "pure" production systems. We then concentrate on the use of PS's in vision, in order to point out both the validity of, and the necessity for, our approach.

2.3.1. Production Systems

The term "pure" production system implies three basic requirements: a set of rules (or productions), a data structure (which is usually some storage area of finite size), and a rule interpreter. In its simplest form, the database is composed of a list of symbols of finite length, with new symbols added on one side at the expense of losing others from the opposite side. The rules are condition-action pairs, with a left hand side consisting of a number of symbols to be matched against the database. If the match is successful (i.e., the symbols are found in the database in the prescribed order), the actions are executed. The latter being the addition at the top of the database of

the symbols specified by the right hand side of the rule. The function of the interpreter in this format is trivial. It matches the conditions and executes the actions. Rules are usually matched in a cyclic order, and the first rule whose conditions match is allowed to fire (execute its actions). Variations on the methods of rule selection and pattern matching led to the design of more complex systems. This will be the subject of discussion in the next sections.

First, we will examine the properties that result from implementing the PS design, and which make it an attractive framework for large AI programs, including image understanding applications. The basic characteristics are summarized as follows:

(1) The database is the sole method of communication between the rules. Each rule can only detect the changes brought about to the database by other rules. The channel is also unique in the sense that all rules have access to all parts of the database [Davis and King, 1975]. Some systems resort to the use of tags or other more complex messages that are deposited by a rule with the purpose of activating another specific rule. This is generally considered against the PS spirit, since it is equivalent to having a rule that invokes another.

(2) The rules are highly independent and modular entities that can be modified, or even deleted, without seriously

degrading system performance [Winograd, 1975]

(3) The price for modularity is in the difficulty of implementing a specific behavior using a finite set of rules (or equivalently explaining that behavior)

(4) Rules represent primitive actions at the conceptual level of the problem. Each action may in fact consist of a complex procedure, as long as its effect is considered to represent a unit of behavior within the context of the system. The conditions are also Boolean combinations of computationally primitive operations that involve only matching and detection

(5) PS provide a unified approach to data and control. The rules have a constrained format that is readable by the system. This allows for a flexible control paradigm in which the system can modify its own rule set

Finally, it should be noted that PS and procedural systems are computationally equivalent, since both can simulate a Turing machine. The selection of a particular approach depends on the requirements of the task at hand. The question is one of efficiency and ease of design and use. Appropriate fields of application of PS will be discussed in section 2.4.2

2.3.2. Rule-Based Systems in AI

The resemblance of PS structure to human cognitive processes motivated the early PS approaches that were concerned, and still are, with modelling human behavior. The database, with its finite length, is analogous to the human short term memory. The designer attempts to formulate a minimal set of production rules that will produce the required behavior. Examples of such systems include PSG [Newell, 1973], a system that was used to explain memory scanning tasks, PAS II [Waterman, 1974] which studied learning and adaptive behavior, and VIS [Moran, 1973] which simulated the descriptive ability of visual images. Some of these systems implemented more sophisticated structures than those of a pure PS (e.g. the use of multiple actions and more complex pattern matching in VIS and PAS II).

Davis and King classify rule-based systems into two categories [Davis and King, 1977]. The first includes the psychological modelling efforts described above. The second class contains what they term performance-oriented expert systems. The latter encode the knowledge about a specific task into productions and attempt to achieve competent behavior in executing that task. The distinction between the two groups goes beyond the differences that motivate them. Large AI programs that are application oriented belong to the second category. They tend to introduce much more complex structures to the rules, the database, and the interpreter. In the remainder of this section, we will discuss examples of

these systems and show how they deviate from the pure form of PS discussed in the previous section

The field of artificial intelligence has gone through the transition from the generality and power of inference in systems like GPS to the more knowledge rich, though methods poor, expert systems [Feigenbaum, 1977]. Psychological evidence supports this trend [Newell and Simon, 1972]. Humans solve their problems by going to the expert, and specialization is the basic theme that organizes human knowledge.

A typical example of a rule-based expert system is the MYCIN medical consultation program [Shortliffe, 1976]. It conducted a dialogue in English with a user (an expert physician), reasoning about a medical case, and ultimately diagnosing and providing treatment for blood infections and meningitis infections. The rules were composed of IF-THEN clauses, and had associated with them "degrees of certainties" representing expert confidence in their validity. MYCIN employed backward chaining of the rules to create a line of reasoning about the case at hand. The conditions of a rule were verified to be either TRUE or FALSE. This might not be possible because of the lack of certain information at one stage during the consultation. In this case, the conditions became the new goals to be instantiated. Several lines of reasoning with varying degrees of confidence were constructed. The decisions rendered by the program agreed with expert opinion in over

90% of the cases tested. More interesting was the system's ability to explain its actions. The various lines of reasoning were available to the program as knowledge sources. This enabled the program to answer questions as to why and how it arrived at a certain conclusion or asked for a specific datum. This property is a direct consequence of the unity of data and control in production-based systems.

DENDRAL, a system that inferred chemical structures for organic molecules, presented a different approach to rule-based expert systems [Feigenbaum et al, 1971]. In this case the data was in the form of graphs whose nodes and links represented atoms and bonds, respectively. The rules embodied theories of mass spectrometry with a situation part representing a particular atomic configuration, and an action part indicating a possible fragmentation to be applied by breaking certain bonds. Thus, the rules were considerably more complex than those in a pure PS. The condition part was actually a whole graph, yet, this was not considered as a major violation of the PS spirit because the rules represented primitive concepts in the domain of the problem [Davis and King, 1977]. DENDRAL was primarily a model-driven system, which employed a generation-and-test heuristic search without feedback. The plausible molecular structures were generated according to the rules. They were then tested and ranked by comparing the generated data to the actual input data using best fit criteria. Performance within very specialized molecular families (for which the program was

taught) proved superior to humans, both in speed and accuracy

The two examples described above have had their offsprings. TEIRESIAS [Davis, 1976] used faulty MYCIN consultations to track and modify, with the help of an expert, the rules that caused the errors. An important addition to this system was the implementation of rules that governed the use of other rules, an idea that will be discussed in more detail in section 2.3.4 because of its relevance to our rule-based vision system. META-DENDRAL [Buchanan et al., 1976] was a system that inferred rules of mass spectrometry for possible later use by the DENDRAL program. The structure was very similar to that of DENDRAL and the resulting rules were compatible to those produced by an expert. The approach underscored the importance of knowledge acquisition for the rule-based systems, and provided an automated framework for such a task.

Other rule-based systems exist in various fields of knowledge. A review of selected systems is presented in [Feigenbaum, 1977], and a recent bibliography on knowledge-based expert systems is given in [Rychenak, 1981]. In the next section, we present an example of a knowledge-based system that did not use rules.

2.3.3. Other Knowledge-Based Systems

There are a variety of other forms in which a knowledge-based system can exist. One prime example that has strongly influenced work in the area, including the design of various rule-based systems, is the HEARSAY-II speech understanding system [Erman et al., 1973, Lesser et al., 1975, Lesser and Erman, 1977]. Knowledge was distributed over a number of independent knowledge sources (KS), each representing the expertise within a particular sub-area of the problem (for speech understanding, these included signal analysis, phonetics, syntax, semantics, and pragmatics). In addition, a global database, termed the blackboard, contained the current state of the system. The basic control mechanism allowed cooperation among the KS's by implementing a hypothesize-and-test scheme. Each KS was able to read and write hypotheses on the blackboard, and this represented the sole method of communication among KS's. The latter is also an important feature of rule-based systems. Other points of resemblance between HEARSAY-II and the rule-based paradigm include the independence of KS's, with each ignorant of which other KS's supplied the information that it used, or which KS's would use the new information it provided. Thus, the KS's resemble the rules, and indeed even their basic structure is compatible. A KS contained three main components: the conditions under which it was activated, the changes it made to the blackboard, and a pointer to the program that executed these changes.

HEARSAY-II departed from the characteristics of rule-based systems in one important aspect. The database (blackboard) was divided into segments that correspond to different levels of abstraction of the data. The KS's were similarly classified according to the level at which they operated. This rendered the database only partially accessible to each of the knowledge sources. Although this was in contradiction with the PS formulation, the data and KS organization allowed for the implementation of a simpler control structure.

HEARSAY-II has affected the design of other expert systems in the field. Both the blackboard approach and the hypothesize-and-test mechanism have been implemented in a number of AI programs (e.g., [Shaheen, 1979] and [Nil and Feigenbaum, 1978]). The hierarchy of knowledge and data is similar in speech and image understanding. Several levels of abstraction exist between the measured signal and the symbolic description that constitutes the output. In HEARSAY-II, different KS's operated independently at different levels, constructing hypotheses about the data at each level. The interaction between the levels was indirect through the hypotheses, with each having "support from above" or "support from below". In the image understanding system, we distinguish between the low level of segmentation and the high level of interpretation. The hypotheses are replaced by regions and lines, and they become the media of interaction between both levels. The low level segmentation rules can be

applied to modify the regions and lines, while the high level rules provide the labels that produce a consistent description of the scene based on the given configuration. Both levels can benefit from the interaction. For example, regions that are not supported by a consistent label can be resegmented by the low level rules. On the other hand, the high level rules will assign interpretations that are based on the confidence of the low level system in the regions.

Having discussed the structure of production systems, and reviewed examples of knowledge-based systems, we now concentrate on an important aspect of the problem, the nature of control in the rule-based methodology. This has been a major cause of diversity in the structure of different rule-based systems.

2.3.4. Control in the Rule-Based Paradigm

The control cycle of the basic PS is a simple mechanism that consists of two phases: a matching phase and an action phase. The former is also known as the recognition phase, and involves selecting a rule whose conditions match the data. The increased complexity of the rules and the data structures of expert systems transformed this rule selection process into a complicated problem that has received much attention and led to a number of proposed solutions. This problem is known as conflict resolution and can be stated as follows: if all available rules test their conditions on the data, there is a possibility that more than one rule will

match The question is which rule to select out of the set of potentially applicable rules (also known as the conflict Rule set)?

The criteria provided for rule selection vary widely [Davis and King, 1977], and include

(1) Implicit order Rules are tested in the same order in which they are stored When a match occurs the process either restarts at the top of the list, or continues from the last rule that fired This is the same as the rule selection mechanism of a "pure" PS

(2) Rule Order All rules are given prespecified priorities, and the rule with highest priority is selected This allows for fast rule matching, but adding new rules becomes more cumbersome

(3) Data Order The rule that matches the most important data item is selected It follows, of course, that all data elements must be ordered The matching process becomes much slower, although rule independent

(4) Generality Order The most specific rule is selected, which is again fast but requires extra work when adding new rules

(5) Temporal Order Selecting the rule that has most recently fired This is also a fast method, but it may be biased towards a subset of the rule set at the expense of using other sources of knowledge

More complicated selection mechanisms have also been proposed For example, DENDRAL used a precedence network to determine the next rule to fire This is a slow process that requires considerable book keeping, both during execution and while adding new rules Rule selection in MYCIN was accomplished by the goal-oriented backward chaining procedure The RHS of rules were scanned, and those that concluded something about the required goal were selected The LHS of these rules were then matched, and if a condition could not be evaluated, it became the new sub-goal, and the process re-iterated The system eventually matched all the rules in the conflict set in order to collect all the data it needed for making its diagnosis

Another method for conflict resolution depends on the introduction of feedback to the PS [Zucker, 1978] The basic idea is that if the PS can compare the effect each of the rules in the conflict set has on the data, it can then select the rule whose actions produce the best outcome The method suggested the use of a relaxation labelling process to select the rule that resulted in the most consistent data set Other knowledge sources must thus be employed to define consistency In our image analysis system, we use feedback by measuring the effect each rule has on a number of

performance parameters. The values of these measures are then dynamically employed to set rule selection and data selection strategies. The availability of this direct feedback path allows the system to detect the consequences of its actions, and perform dynamic adjustments accordingly.

A modification to the rule selection method used by MYCIN was introduced by Davis in TEIRESIAS. The system used higher level rules that decided, based on the data, which group of rules was more suitable for the task at hand [Davis, 1976]. This set of control rules, were termed meta-rules, since they were actually rules about rules. Although their presence would violate the PS property of not allowing rules to invoke others, Davis rightfully argued that the spirit of modularity was left untouched. This was because the meta-rules referred to rule content and not to specific rules by name, and therefore the addition of new knowledge rules was taken care of automatically. A rule hierarchy can thus be constructed with the knowledge rules at the lowest level, followed by meta-rules that specify rule selection strategies. Higher levels are also possible, with strategy rules that specify which rule selection strategy to select, and so on. Later, the idea of meta-rules was generalized for all knowledge-based systems by advocating the creation of meta-levels of knowledge [Davis, 1980].

The advantages of using this formulation can be stated as follows

(1) It provides an intuitive solution to the rule selection problem

(2) It allows the designer to "reason about the control" of his system. In fact, finding the proper control strategy itself can be posed as a dynamic problem to be solved by the same AI system

(3) The mechanism used for reasoning at the object level is the same as that used at the control level.

(4) The control structure is explicitly represented by the meta-rules, just as the knowledge is, and thus the system behavior becomes more visible and more explainable

In the rule-based vision system presented here, the control strategy implemented is similar to that described above, with the notable addition of two features: the creation of a second type of control rule and the inclusion of a third higher level in the rule hierarchy.

The first addition results from the nature of the problem at hand. Image processing systems have to deal with a comparatively large amount of data (measured and inferred). The order in which the data is matched affects the final outcome and is reflected in the spatial order in which an

image is processed. For the rule-based system, this data selection problem exists in parallel with the rule selection problem. The same argument that advocated the inclusion of meta-rules to reason about rule selection is used here to create a new set of control rules that reason about data selection. The latter are termed focus of attention rules because they have actions that select the next data item to match the knowledge rules on. They exist at the same conceptual level as the meta-rules, and are used in conjunction with them to create various processing strategies. The focus of attention rules will be described in detail in chapter 5.

The second new feature is related to the selection of different processing strategies for the system. As indicated above, a particular set of meta-rules and focus of attention rules will correspond to a specific strategy. A set of higher level rules, termed strategy rules, are used to reason, based on the data, about the suitable strategy to advocate at any point in processing. The three levels of rules were illustrated in figure (1.2). This notion of dynamic strategy selection will be discussed in detail in chapter 6.

2.4. RULE-BASED SYSTEMS FOR COMPUTER VISION

Computer vision is currently one of the most active fields of AI applications. The psychological, physiological, and functional aspects of the visual task are being thoroughly studied and analysed [Levine, 1993]. It is hoped that this approach will lead to a better understanding of the biological process, and hence aid in the creation of a computer vision system that simulates the functional behavior of the eye-brain system. Rule-based systems have been used to simulate human performance in a number of psychological tasks. One of the main goals of this thesis is to study the usefulness of the rule-based paradigm in solving one aspect of the vision problem. Compared to the general approach undertaken in this research, previous efforts along the same line have been quite limited, and will be reviewed in this section. This will be followed by an analysis of the credibility of using a rule-based system for image understanding.

2.4.1. Review of Rule-Based Vision Systems

Until a few years ago, the use of production systems had been totally ignored by the designers of scene analysis systems [Duda et al., 1978]. Since then, few attempts have surfaced that are worth studying. The following discussion provides insight into the shortcomings and the problems encountered by these approaches.

One of the early attempts was not actually in image understanding, but rather studied the symbolic representation of visual images by humans [Moran, 1973]. In a system called VIS, Moran compiled a set of rules that simulated the behavior of a human describing a sequence of simple visual events. A human subject was blind-folded and instructed to move an imaginary token successively in unit steps in one of four directions. His verbal response in describing the created path was recorded. The program took the directions as input, and used its rules to produce similar descriptions of the unfolding path.

In order to bypass the complexity of unstructured programming of production systems, the rules were grouped into procedures, within which they were totally ordered. Each procedure had a predefined task, and the transfer of control between procedures took place according to a standard flowchart. The rule selection mechanism was thus implicit and simple, but this was at the expense of compromising rule modularity and independence. The significance of this system is that it demonstrated the difficulties encountered in designing the control structure of a PS, even for a simple AI task that requires very little computation. The author noted the need for "a control language" that encodes "precise statements of the relations between rules so that any rule can fire sensibly in a variety of contexts". Later in this dissertation, we will describe a mechanism that unifies the representation of knowledge and control. Different control

strategies will be expressed in terms of combinations of control rules, thus providing a language that specifies strategies for the much more complicated task of low level image analysis

The first attempt at using production systems in image understanding was reported in K R Sloan Jr's thesis [Sloan, 1977]. The effort was restricted to the high level control part of the system, and was clearly separated from the low level image analysis part. A simple region grower was used to produce a region map from which the color, shape, texture, and position features of all regions were computed. This "image processing stage" was then interfaced to the high level system that used a world model to assign interpretations to the regions. Control was transferred sequentially from the low level to the high level stages without feedback. The rule-based structure was limited to the interpretation stage. The control mechanism implemented a production system that was originally developed for natural language processing [Joshi, 1978]. The latter employed a novel approach to matching that is based on allowing partial matches, so that the rule with the "best" match was allowed to fire. In addition to data modification, rule actions contained procedures that modified other rules. Knowledge about the class of scenes to be analysed was coded into the rules. A subset of natural outdoor scenes with no man-made objects and containing a limited number of recognizable objects (e.g. sky, trees, and mountains) was used.

The results were moderately good for the class of images tested. Errors produced by the simple region grower were not recoverable because of the absence of top-down feedback. Sloan recognized the lack of "more powerful and more refined" low level processing for his system. Our low level image analysis system was partly motivated by such deficiencies (see [Shaheen, 1979] for another image interpretation system that cited the need for better low level segmentation). The introduction of the rule-based approach to the low level stage establishes a common ground of interaction within the image processing hierarchy that could not be achieved in Sloan's system using primitive low level operators. Rules at both levels can communicate through the production database, allowing both bottom-up, as well as top-down processing.

Once again, rule selection posed difficulties for this image interpretation system. Originally, the production system partially ordered the rules according to specificity. The interpreter selected the rule whose conditions were best satisfied by the data. If no single rule existed, the set of rules which together covered most of the data conditions was chosen. If two or more of these were incompatible, the selection was made at random.

Because of the computational complexities imposed by the application to image analysis, the above mechanism had to be modified. The number of eligible rules at any time had to be trimmed down in order to achieve reasonable computation efficiency for the amount of data involved. Sloan introduced

direct control over the rules, by allowing the actions of some rules to "ground" other rules (prevent them from being tested). This made it possible to choose the set of rules that is most suitable within a given context. However, direct control reduced the freedom of expression provided by the PS because it seriously diminished modularity. The author noted that "it would be desirable to have an automatic method of structuring rules which provided the necessary focussing without limiting learning". The system presented here possesses a dynamic rule ordering capability that does not compromise rule modularity. The use of meta-rules to specify different control strategies represents a more structured approach to rule organization than the direct control imposed in Sloan's system. In chapter 6, we will describe the details of this rule selection mechanism, and later, in chapter 7, we will show how it enhances the rule learning process.

The final relevant aspect of Sloan's system is that it demonstrated the need for a mechanism to deal with the large amounts of data associated with image processing. A buffer memory was introduced that only stored facts and assertions that were currently relevant. All other assertions about the scene under analysis were stored in a larger long term memory. The disadvantage of this method is that it required a lot of bookkeeping to store and retrieve information between both memories, while deciding the relative importance of each data item. The method also introduced another type

of direct control, since the rules were selectively allowed access to the data. Our rule-based system deals with the large data constraint by introducing a modular focus of attention mechanism that spatially selects the next data item to match the rules on. The process itself is rule-based and data-driven, and the focus of attention strategy is set in a manner that is analogous to the rule selection strategy.

Bullock has discussed two image analysis systems that have been constructed using production systems [Bullock, 1977]. The first was a high level system that employed simple graph rules to derive relational information from the database in response to simple queries. The image was first segmented into regions that had uniform texture properties [Kirsch, 1971]. The position, size, and spatial relations of the regions were then stored in a tree structure and made available to the high level system. The basic structure is thus the same as in Sloan's system, where direct sequential flow of control was advocated between the low and high level stages of the analysis. The use of production rules was again deferred to the semantic level of processing, thus restricting the low level analysis portion into the standard procedural approaches that have considerably less computational and inference powers. In addition, this structure allowed for a very limited channel of interaction between the different stages of the image understanding system.

The second system described by Bullock was a low level system that simulated edge, line, and corner detectors. The method suggested that the image be transformed into a symbolic pixel array [Tanimoto, 1977], where pixels have pointers to collections of descriptive tags (e.g., intensity, edge point, vertex, etc.). The rules would then have condition sides that are also symbolic pixel arrays depicting different situations, much in the spirit of the molecular graphs in DENDRAL [Feigenbaum et al., 1971]. Since the tasks attempted by this system were inherently procedural (e.g., simulating edge mask operators), production rules proved to be computationally inefficient. This lead Bullock to conclude that "there should be a split in uniformity so that the low level operators are written as procedures and the higher level operators are written as production rules".

The above conclusion is partly justified. There is nothing gained by implementing procedural algorithms in production form. However, the argument that all low level operations are naturally procedural is perhaps hasty. Procedural operators, such as edge detectors and region growers, compute initial configurations from the image. A considerable amount of low level processing must still be applied to the output of these operators before it becomes ready for the interpretation stage. Previous approaches have dealt with this problem by applying various heuristics to clean-up the output. A large number of criteria has been developed to bridge the gap between the low level operators

and the high level analysis [Riseman and Arbib, 1977] In this thesis, the rule-based approach is advocated because it provides a formal mechanism for implementing these criteria

ACRONYM is a model-based vision system that employed a rule-based reasoning component [Brooks et al, 1978] Rules were used to generate specific instances of objects from their generic descriptions The former were then compared to actual data in order to locate the objects in an image The basic control structure implemented MYCIN's consequence-driven methodology In another rule-based system sequences of images were analysed for cell-tracking applications [Youssef, 1982], and rule order was determined in a procedural form similar to VIS

All the approaches discussed above have concentrated on high level aspects of the vision problem The rules represented specific domain knowledge that is application-dependent What is needed is a representational mechanism that can encompass general purpose knowledge about image formation and the laws of perceptual grouping These information sources are independent of the specific application⁶ and the semantic content of the images Such a formulation fills the gap between the procedural operators that measure and detect features in the image and the high level expert systems that interpret the results Chapter 5 discusses the structure and type of low level knowledge implemented by the system

2.4.2. The Rule-Based Approach to Segmentation

Production systems may not always be suitable. For some applications a procedural approach might be more adequate, and in others the PS formulation is awkward. Waterman has illustrated the complexity of using his PAS II system for the addition of positive integers, a trivial effort in procedural form [Waterman, 1974]. Davis and King have investigated the characteristics of appropriate and inappropriate domains for PS applications [Davis and King, 1977]. We will use their conclusions to study how well the low level vision problem conforms with PS requirements.

A system that is composed of many independent states is more suitable for PS formulation than another that is based on a single unified and well defined theory. Image segmentation is a good example of the first type because of the richness of the information present in the input image. The independent variables include spatial position, grey level intensity (or color), and factors that influence image formation (e.g., shadows, highlights, projections, and digitization effects). The factors that constrain the relation between these variables are in the form of loose concepts, for example, uniformity, similarity, continuity, and the presence of texture. These highly independent and unconstrained characteristics make it difficult to imagine a solution that is based on concise and well defined principles.

Control for PS is simple and extremely flexible. This makes it best suited for applications that consist of a set of independent actions as opposed to a multitude of interconnected processes with attached subprocesses. The image segmentation task is divided into a number of independent actions that merge and split regions, and others that add, delete, extend and connect lines. No direct communication is required between any of these actions as they cooperate and compete to produce the final output. Communications between the different processes are thus at a minimum, and modularity is considered an important asset in the design and functioning of the system.

A third PS eligibility requirement is the ability to separate the knowledge from the control structure. The concepts used for image analysis are best suited for coding into production rules. This is because the criteria for region grouping and line formation can be easily stated in rule form, irrespective of the way they are used in the analysis. An example of this, is in stating the fact that adjacent regions that have similar properties should be merged. The conditions posed and the action cited are independent of when, where, or how the event takes place. This is contrasted to the case where a procedure describes how to accomplish a certain goal.

Image segmentation is a task that requires a broad scope of attention with the ability to focus on and react to small changes in the data. An image may contain a large number of unrelated data configurations that require the attention of a variety of independent knowledge sources. Each data modification established by one of the knowledge sources incrementally contributes to the overall outcome. The factors analyzed above have strongly motivated the choice of the rule-based solution to the low level segmentation problem. In chapter 4, we will present additional structural constraints that rendered this selection an attractive one.

Having discussed the suitability of the low level vision problem to the PS formulation, we now describe the degree by which the system presented here is consistent with PS properties. The main aspects can be summarized as follows:

(1) The basic components of PS architecture are included, namely, the database, the set of rules, and the rule interpreter.

(2) As with all other rule-based AI programs, the database is not just a string of symbols. Instead, it is a large storage medium that contains the states of the system. The latter represent the input data, as well as the segmentation data and the final output.

(3) The rules are in the form of condition-action pairs. The

conditions are simple logical predicates that require the examination of the data, and return only an indication of the success or failure of the matching operation. The actions are independent conceptual primitives for the segmentation problem. Each modifies one or more of the states in the database through the application of an elementary operation. These include the merging and splitting of regions, as well as the addition, deletion, extension, and connection of lines.

(4) The rules are the modular units of knowledge. They can only communicate with each other through the effect they have on the database. This property is particularly important in building the world model. Sources of knowledge can be added, deleted, or modified independently. This makes the learning process considerably simpler, and allows for automatic updating of the model.

(5) The database is not partitioned, so that all rules have access to any of the stored data items. Although the size of the database is large, incremental changes in the data are broadcast to all knowledge sources.

(6) Control is achieved through the use of multiple levels of rules. Besides the standard knowledge rules, two types of control rules are introduced. A rule selection strategy is programmed through the use of a set of meta-rules, and a data selection strategy employs a set of focus of attention rules.

A third level of strategy rules allows for the choice between different rule selection and data selection strategies

(7) The rule hierarchy described above does not compromise modularity, since the rules at any level refer to rules at a lower level by their content and not by their specific names. For example, a meta-rule might indicate that it is preferable in the next cycle to match rules that modify regions and not those that modify lines. It does not specify the firing of a particular region rule, and hence the effect on adding or deleting rules at a certain level is minimal.

(8) This multi-level rule hierarchy produces a control structure with the following advantages.

- (i) An increased speed, because not all rules have to be matched in each cycle.
- (ii) An explicit control strategy that is accessible to the system designer, as well as the processing modules.
- (iii) The possibility of creating a dynamic strategy selection mechanism.

This chapter has dealt primarily with the relationship between computer vision as an application and rule-based systems as a processing mechanism. The field of computer vision, and in particular, low level image processing was briefly examined. Previous rule-based systems in other applications, as well as in computer vision were reviewed. Finally, the motivations for using the rule-based approach to

the problem at hand were presented. This was followed by a brief description of the major properties of the low level vision system presented here, from the point of view of its consistency with basic PS properties. The remaining chapters describe in detail the structure and mechanisms of the system

CHAPTER 3

THE DATA STRUCTURE

3.1. INTRODUCTION

The computer vision problem can be defined as a transformation of data from one representation (the measurements in the input image) into another (a symbolic description of the image). This transformation is not a direct one. Instead, the data passes through intermediate stages that correspond to successive levels of visual abstraction. In chapter 2, we described the representational hierarchy that has been established for image understanding systems. Different levels of description include pixels, regions, lines, surfaces, and objects. Low level segmentation is generally concerned with the first three data types, while the higher level representations are used by the interpretation process.

The data for rule-based segmentation can be classified into two disjoint subsets. The first contains information that is specific to the image under analysis. This includes the measured input, the data generated during processing, and finally the output. The second type of data represents knowledge that is independent of the particular image. This knowledge is coded into the production rules and constitutes the world model used by the system processes. We have distinguished between the two types of data by storing them in separate databases. The short term memory (STM) is the storage medium for the image specific data. Its contents can be accessed and modified by any of the processing modules. STM data lose their significance as soon as the current image is "forgotten", or equivalently, when processing moves to another image. In contrast to the above, the long term memory (LTM) contains the world model which is also accessed and changed by the processing modules, but whose contents represent the current knowledge (initial and acquired) that is used to analyse all images.

Chapter 5 is dedicated to the LTM and discusses the low level processing model represented by the production rules. The data stored in the STM are the subject of this chapter, and are classified into three major categories: the static image data, the dynamic segmentation data, and system information and statistics. The first class contains the input image measurements, as well as other computed data that do not depend on the state of the segmentation, such as the

gradient of intensity at each pixel in the image. The segmentation data describe the regions, lines, and areas of attention deduced by the processing modules. Two aspects are of special interest here: the integration of regions and lines into the same data structure, and the introduction of focus of attention areas. The final class of data includes all the computations that are stored in the STM for control strategy determination, as well as for learning purposes. Before we proceed to discuss the three data types in more detail, a brief description of the database model and the notations used is in order.

3.1.1. The Database

Both the STM and LTM have been implemented as relational databases [Codd, 1970]. Shaheen discussed the suitability of this representation for storing image data and compared it with hierarchical and network models [Shaheen, 1979]. The main advantage of using the relational model is that it deals with data entities and relations between them in a uniform manner. In general, databases associated with image processing have to deal with large amounts of data. A simpler structure results in simpler accessing mechanisms and easier user interaction.

The basic storage unit is a relation which is defined on a finite number of domains $D_1, D_2, \dots, D_n$. Information is stored as a set of n -tuples $(d_1, d_2, \dots, d_n)$ such that d_1 belongs to D_1 , d_2 belongs to D_2 , etc. The relations defined

in this chapter and the next will be represented in terms of their domains as follows:

$$D_1, D_2, \dots, D_i \xrightarrow{\quad\quad\quad} D_{i+1}, D_{i+2}, \dots, D_n \quad (3.1)$$

where the first i domains are the key domains of the relation that uniquely define each n -tuple (i.e., each new tuple must have a different value for at least one of the first i domains). It is convenient to visualize a relation in the form of a table with each row representing an n -tuple and each column containing the values for one domain. Table (3.1) contains a relation on three domains, the first of which is a key domain, so that

$$\text{Name} \xrightarrow{\quad\quad\quad} \text{Age}, \text{Weight} \quad (3.2)$$

In general, the simplicity of this database organization minimizes redundancy and facilitates access and modification of the various data items. This is an important requirement for the rule-based system, because of the highly frequent and repetitive "match and modify" operations. The next sections describe in detail the data stored in the STM.

Name	Age	Weight
John	25	165
Paul	12	120
Helen	19	135

Table (3.1) : An example of a relation

3.2. THE STATIC IMAGE DATA

As the name implies, these data do not change with processing. They represent features of the image (measured or computed) at the pixel level. Therefore, these features are independent of how the image is partitioned. Included in this category are the intensity measurements of the input image and their first derivative.

3.2.1. The Input Image

The measured data are in the form of one or more two-dimensional arrays of values with m rows and n columns. Each array represents a feature of the image, the most common of which are the intensity levels at each pixel (array element). For black and white images, a single value is measured that corresponds to the grey level intensity. Color images, on the other hand, require the storage of three arrays that contain values for the intensity of each of three color channels at each pixel. In most input devices, standard filters are used to separate intensity values for the red, green, and blue channels. An example of a digitized color image is shown in figure (3.1).

The analysis presented here does not rule out the use of other measurements from the scene. For example, such additional information can come from range data that indicate depth information for various objects at every pixel. In general, M arrays are read into the STM by the input

mechanisms, where $M = 1$ for black and white images, $M = 3$ for color images, or $M > 3$ if other measurements are included. The input arrays are stored in M FEATURE relations that have m rows of $(n+1)$ -tuples of the form:

ROW NO	---	C_1	C_2	...	C_n	(3.3)
		1	2		n	

The last n domains represent the vertical columns of the image, while the tuples represent horizontal rows. This one-to-one correspondence between pixel values and entries in a relation contributes to the efficiency and ease of using this data structure. Equation (3.3) depicts the format used for storing all other data computed at the pixel level, such as gradient information and region and line labels, as described in the following sections.

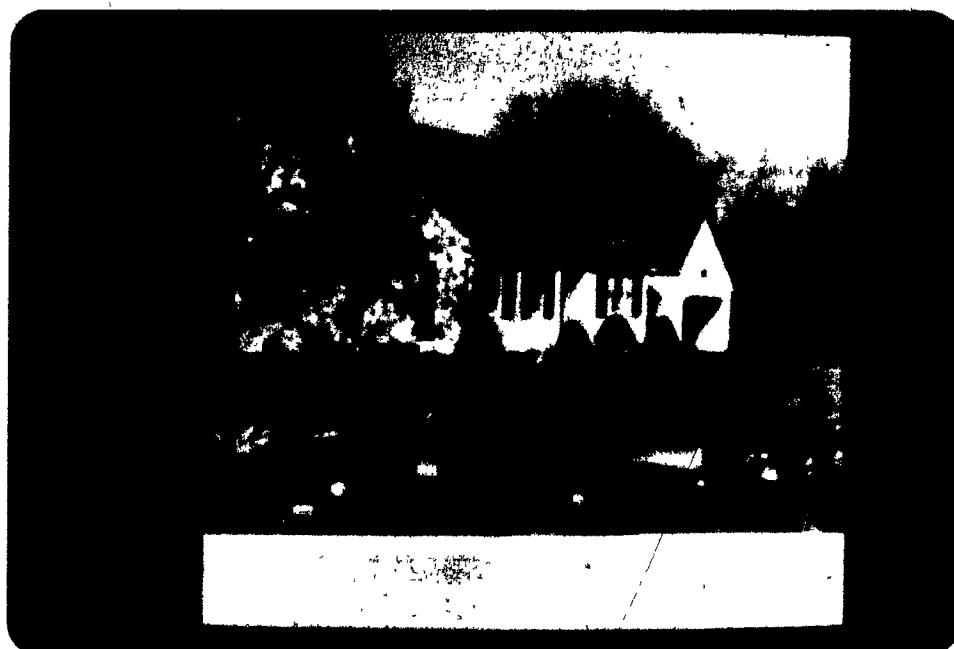


Figure (3.1) . Digitized color image of a natural scene.

3.2.2. Gradient Information

Because of the importance of discontinuity information in detecting and processing lines, and in determining the uniformity of regions, the first derivative of the measured features is computed and stored in the STM. The two-dimensional gradient is evaluated at each pixel by convolving the image feature arrays with an array of values representing the weights associated with one of the gradient mask operators. An example of these is the 3X3 Sobel operator shown in figure (3.2). The convolution of this mask

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figure (3.2) : The Sobel Gradient Operator

with all measured features will result in M values at each pixel, and a weighted sum is taken to represent the magnitude of the gradient. The direction at which the gradient is a maximum is computed by rotating the mask into each of the eight possible positions shown in figure (3.3). The mask that produces the maximum gradient magnitude is selected at each pixel, and the number associated with that mask represents the gradient direction at that pixel. The direction codes are chosen to be compatible with the chain coding scheme for boundary representation [Freeman, 1961], shown in figure (3.4a). The convention used is that the edge

element at a pixel is taken to be perpendicular to the gradient direction, and has an arrow that points in such a way that the brighter side of the gradient is to its left, as shown in figure (3.4b).

1 2 1 0 0 0 -1 -2 -1	2 1 0 1 0 -1 0 -1 -2	1 0 -1 2 0 -2 1 0 -1	0 -1 -2 1 0 -1 2 1 0
[0]	[1]	[2]	[3]
-1 -2 -1 0 0 0 1 2 1	-2 -1 0 -1 0 1 0 1 2	-1 0 1 -2 0 2 -1 0 1	0 1 2 -1 0 1 -2 -1 0
[4]	[5]	[6]	[7]

Figure (3.3) : Directional masks for the Sobel operator.

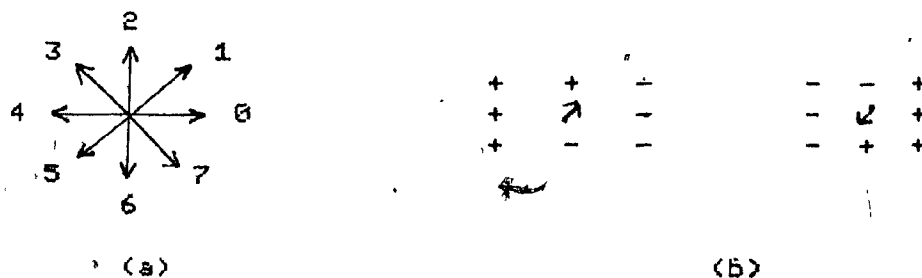


Figure (3.4) : Representing gradient direction.

(a) The chain code direction scheme.

(b) The convention used for gradient direction.

Gradient information contributes two more image arrays to the STM, one for magnitude and the other for direction. They are stored in two gradient relations that have the same format given in equation (3.3). This information is used by

the processing modules in creating and modifying the image segmentation, as will be discussed in detail in chapter 4. The gradient magnitude and direction images for the scene of figure (3.1) are shown in figure (3.5).

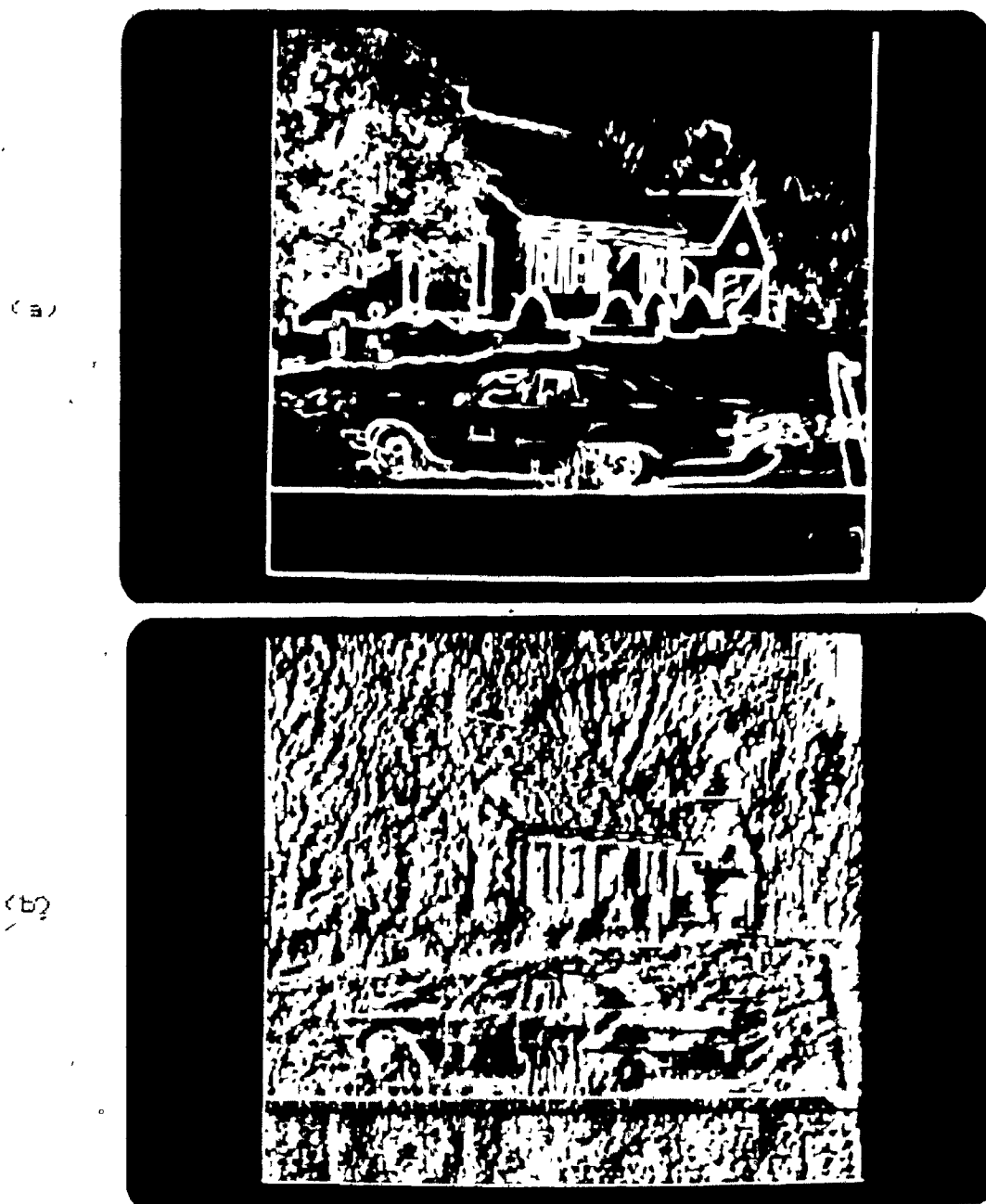


Figure (3.5) Gradient information for scene in figure (3.1)

(a) The gradient magnitude image

(b) Gradient direction coded into 8 grey levels

3.3. THE SEGMENTATION DATA

The image partition is described by three basic entities. the regions, the lines, and the focus of attention areas. Following are definitions of these, their representation, their stored features, and the relations among them

3.3.1. Regions

The image is divided into regions by attaching labels to the pixels so that each label defines the particular region number to which a pixel belongs. The labelling scheme used has the following properties:

(1) Each pixel in the image must have a label, i.e., the segmentation is complete in the sense that it accounts for all parts of the image. This is in contrast to other segmentation outputs that identify particular objects against a background of non-partitioned pixels (e.g., detecting blood cells in microscopic images).

(2) A pixel has exactly one region label. This implies that regions are not allowed to overlap in the image.

(3) A region is composed of pixels that are four-connected, so that it is always possible to travel between any two pixels having the same label without visiting pixels belonging to other regions.

(4) There is no restriction on region size other than the resolution of the image. It follows that a region can be composed of a single pixel, or it can include all the pixels in the image.

The region partition is best represented by the region map relation, which follows the format in equation (3.3). It is an image array in which the value at each pixel (or entry in the relation) is equal to the label of the region to which the pixel belongs. The reader will note that the region map preserves all the spatial relations among regions, as well as the position and shape features. An example of a region map that contains five regions is shown in figure (3.6). In this thesis, region partitions will be demonstrated by superimposing the boundaries on the original image. Figure (3.7) is an example of hand-segmented regions for the image in figure (3.1).

1	1	1	1	1	1	2	2	2	3	3	3
1	1	1	1	1	2	2	2	2	3	3	3
1	1	1	1	2	2	2	2	2	3	3	3
1	1	2	2	2	2	3	3	3	3	3	3
2	2	2	2	2	2	2	3	3	3	3	3
2	2	2	4	2	4	2	2	2	3	3	3
2	2	4	4	4	4	4	4	5	5	3	3
2	4	4	4	4	4	4	5	5	5	5	3
2	4	4	4	4	4	5	5	5	5	5	5
4	4	4	4	4	4	5	5	5	5	5	5

Figure (3.6) : An example of a region map.



Figure (3.7) A partition of the image into regions

- The following sections define the features that are explicitly computed and stored for further use by the processing modules

3.3.1.1. Descriptive Features

These can in turn be classified into:

(A) Statistical Features

The measured features of an image are represented by M image arrays, as described in section 3.2.1. The average value of each feature over all the pixels belonging to the same region indicates the brightness level of that feature across the

region (average gray level intensity or color). Deviations from the average value are best represented by the variances of the features, which are also good indicators of the uniformity of the measurements over the region. Let the value of feature F_k at each pixel i belonging to region R_j be f_{ki} , then the mean of these values $\bar{f}_k$ is given by

$$\bar{f}_k = \frac{\sum_{i \in R_j} f_{ki}}{A_j} \quad (3.4)$$

where A_j is the area of region R_j . The variance of feature F_k over region R_j is then given by

$$\sigma_k^2 = \frac{\sum_{i \in R_j} (f_{ki} - \bar{f}_k)^2}{A_j} \quad (3.5)$$

Equations (3.4) and (3.5) are also used to compute the mean $\bar{g}$ and variance σ_g^2 of the gradient array over a region. Note that the average value of the gradient provides an additional measure of region uniformity.

(B) Position Features :

In order to achieve quick access to the information stored in the pixels that belong to a particular region, pointers are needed to its spatial position within the image array. The centroid coordinates (x_c, y_c) are computed by averaging the

horizontal and vertical coordinates of all pixels (x_i, y_i) belonging to a region R_j . Thus,

$$x_c = \frac{1}{n} \sum_{i \in R_j} x_i \quad \text{and} \quad y_c = \frac{1}{n} \sum_{i \in R_j} y_i \quad (3.6)$$

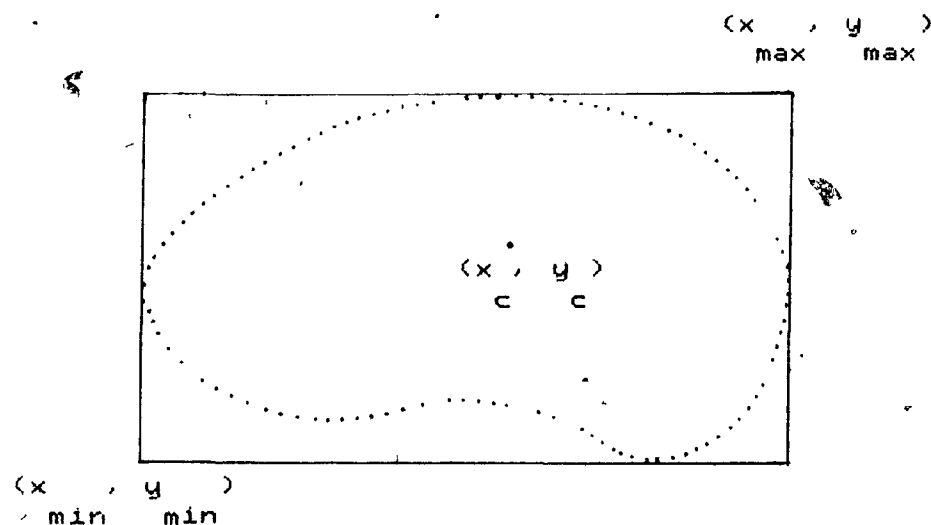


Figure (3.8) The centroid and minimum rectangle of a region.

In addition, the coordinates of the minimum bounding rectangle $(x_{\min}, y_{\min})$ and $(x_{\max}, y_{\max})$ shown in figure (3.8) are also stored. The rectangle whose sides are horizontal and vertical is chosen over the more accurate rectangle that has sides parallel to the major and minor axes of a region. This is because the former is computationally less expensive and provides the required position information with reasonable accuracy. Position features prove to be very useful in locating a region, since the alternative is to scan the region map array searching for the region label.

(C) Shape Features

Simple features are included that give an indication of the approximate shape of a region. An estimate of the size is given by the total number of pixels that possess the region label, also referred to as the region area A .

The perimeter of a region is equal to the length of its boundary. A popular measure of the perimeter is calculated from the chain code representation of the boundary [Shaheen, 1979]. A unit measure is added for vertical and horizontal increments along the boundary, while a factor of $\sqrt{2}$ is added for diagonal steps. However, this method for computing the perimeter becomes inaccurate for small region sizes. For example, the perimeter for a region that is composed of one pixel is not defined, while that of the two-pixel region shown in figure (3.9a) is equal to 2. A new and more appropriate estimate is used here and will be shown to be better suited for computing adjacency between regions. The value of the perimeter P is taken to be equal to the number of pixels that are four-connected neighbors of the pixels on the region boundary, each counted once, as shown in figure (3.9b).

In addition, two other shape features are computed, the circularity θ and the aspect ratio δ . The former estimates how different the shape of the boundary is from that of a circle, and is given by

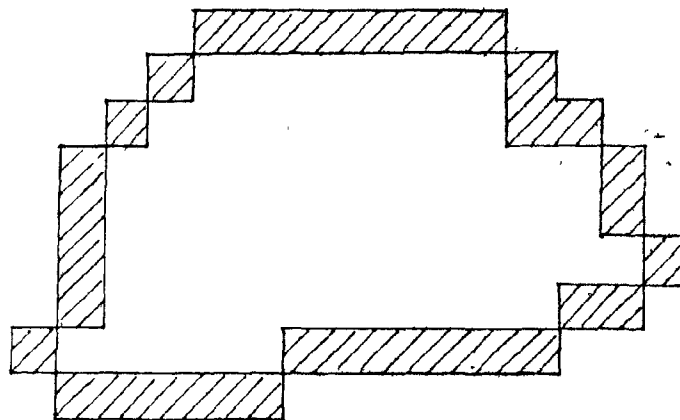


$$\begin{aligned} cc &= 0 \\ P &= 4 \end{aligned}$$



$$\begin{aligned} cc &= 2 \\ P &= 6 \end{aligned}$$

(a)



(b)

Figure (3.9) Perimeter computation.

(a) Failure of the chain code method for small regions

(b) Perimeter using four-connected neighbors.

$$\theta = \frac{4 \pi A}{P} \quad (3.7)$$

The aspect ratio is defined as the ratio of region height to width, and is computed as follows

$$\delta = \frac{y_{\max} - y_{\min}}{x_{\max} - x_{\min}} \quad (3.8)$$

The descriptive features of the region are stored in the region description relation, given by

$$\begin{aligned} \text{Region No } (j) \longrightarrow & \bar{f}_1, \bar{f}_M, \bar{c}_1^2, \dots, \bar{c}_M^2, \bar{g}, \bar{c}_g^2, \\ & (x_c, y_c), (x_{\min}, y_{\min}), (x_{\max}, y_{\max}), \\ & A, P, \theta, \delta. \end{aligned} \quad (3.9)$$

Another estimate of the region position is given by the chain code representation of its boundary. The coordinates of one boundary pixel are stored, along with a description of the path that starts at that pixel and follows the boundary. Because a region is composed of connected pixels, the boundary must form a closed path that will return to the starting pixel. Only the incremental directions from one boundary pixel to the next need to be stored. These are coded according to the notation given in figure (3.4a), and are stored in the region chain code relation as follows:

Region No. $\rightarrow$ x_s, y_s , Length, Chain Code, (3.10)
 $s \quad s$

where (x_s, y_s) are the coordinates of the starting point, and Length is equal to the number of boundary pixels. The chain code representation provides a fast method for traversing a region boundary. This proves to be very useful in computing spatial adjacency with other regions, as explained in the next section.

3.3.1.2. Spatial Features

The low level processes require the knowledge of which regions are adjacent, as well as an estimate of the relative amount of adjacency between each pair of regions. This information is not only useful for analysing each region, but also in determining a strategy for moving from one region to the next during processing. These ideas will become apparent

In later chapters

The value by which region R_j is adjacent to region R_i is defined as the ratio of the perimeter shared between the two regions P_{sh} to the total perimeter P_i of region R_i [Shaheen, 1979]. P_{sh} is equal to the length of common boundary, or in other words, the number of pixels belonging to region R_j that are four-connected to pixels belonging to region R_i , as shown in figure (3 10). Let the adjacency value between the two regions be $a_{i,j}$, then

$$a_{i,j} = \frac{P_{sh}}{P_i} \quad (3 11)$$

Therefore, the sum of adjacency values for all regions adjacent to R_i is one

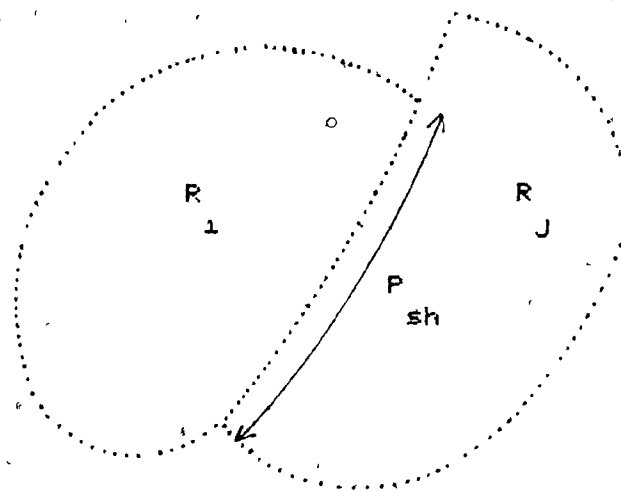


Figure (3 10) Spatial adjacency between regions

This concludes our discussion of the STM region features. Table (3.2) is a sample of the information computed and stored for one region, namely, the grass area in the partition of figure (3.7).

Information on region 26

Average color features (RGB) :	36.59	41.17	24.38
Variance of color features :	4.77	3.91	4.45
Average gradient :	14.4266529		
Bounding rectangle :	(2, 62) and (255, 127)		
Centroid coordinates :	(158, 95)		
Area :	4690		
Perimeter :	743		
Circularity :	0.11		
Aspect Ratio :	3.85		

Adjacent regions (10) are:

22 --> 0.21	28 --> 0.17	15 --> 0.11	31 --> 0.09
24 --> 0.09	32 --> 0.08	30 --> 0.05	55 --> 0.04
19 --> 0.04	29 --> 0.03		

Table (3.2) : Example of the features stored for a region.

3.3.2. Lines

The gradient information derived in section 3.2.2. is used to guide the initial formation of lines, as well as subsequent modifications during rule-based processing. Lines are defined as separate entities that do not necessarily correspond to the boundaries of the regions described in the previous section. The basic assumption is that since lines and regions are derived from independent sources (discontinuity for lines versus uniformity for regions), they carry different information, and therefore their simultaneous

use should improve the partitioning of the image produced by either method separately. This is accomplished by employing lines in region analysis, and using regions to guide line processing.

A second motive for implementing lines separately is that they do exist as distinct objects (rather than boundaries of objects) in images of real world scenes. The image of a telephone line against a sky background is a good example of such occurrences. The shadow of such a line is visible on the grass behind the car in the scene of figure (3.1). A representation in terms of regions is often ambiguous and sometimes impossible. Of course, such lines can be visualized as thin objects, but because they approach the limits of the resolution of the human eye (and the digitized image), the notion of a boundary for such objects becomes physically unrealizable. Furthermore, region analysis - which is based on optimizing feature uniformity - will in most cases overlook the presence of such abrupt discontinuities, especially if they take place within large uniform areas.

Structurally, a line consists of a group of eight-connected pixels that has the form of a flexible filament which can assume any shape. The method by which lines are initially traced from a group of edge points will be described in chapter 4. Each line is identified by a specific label that is attached to each pixel belonging to that line. An image array, termed a line map, describes the

lines within an image and has the following properties

(1) A pixel may belong to a line, and therefore bear its label, or it may not be part of any line. In the latter case, the pixel carries a null (or no-line) label. Thus, by nature, the line map is not complete in the sense described for the region map in the previous section

(2) Lines are composed of connected pixels so that it is always possible to form an eight-connected path which starts and ends on any two pixels, and is made entirely of pixels belonging to the same line. Therefore, if two line segments are not connected, they will bear different line labels, even if they are part of the same physical boundary.

(3) Each pixel can carry one line label at most, which implies that no overlap is allowed between lines. Therefore, we may associate with each pixel in the original image exactly one region label and one line label.

The line map is stored in the STM in a relation of the form given by equation (3.3). Because most of the pixels in an image do not belong to lines, this method of storage may seem inefficient. However, the inclusion of a line map is essential for computing spatial relations among lines, as will be described later in this section. Figure (3.11) represents a line map for the scene in figure (3.1) in which the lines are superimposed over the original image

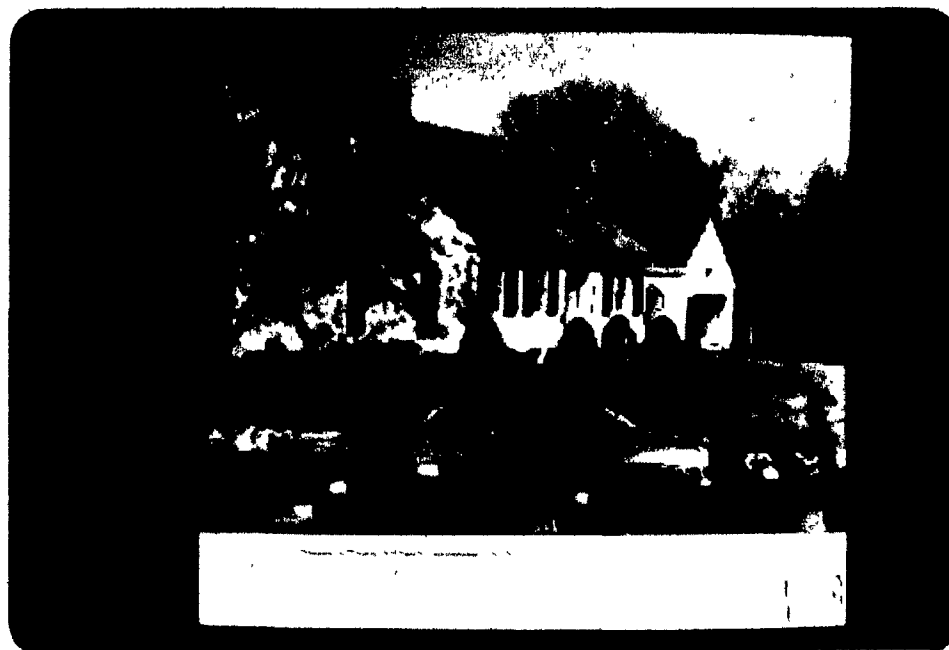


Figure (3 11) An example of a line partition

3.3.2.1. Descriptive Features

Various line features are computed and stored in the STM. Some are used by the line analysis process to verify the conditions of different rules. Others provide means for fast and easy access to the information carried by a line.

(A) Statistical Features

The mean and variance of each of the M measured features at pixels belonging to the same line are computed in the same way as for regions. Thus, if f_{k1} is the value of feature F_k at pixel 1 of line L_j , then

$$\bar{f}_k = \frac{\sum_{i \in L_j} f_{ki}}{l_j} \quad (3.12)$$

$$\text{and } \sigma_k^2 = \frac{\sum_{i \in L_j} (f_{ki} - \bar{f}_k)^2}{l_j} \quad (3.13)$$

are the mean and variance of that feature, respectively, and l_j is the length of line L_j .

The mean and variance of the gradient for the pixels of a line are computed in a similar manner. Note that the average magnitude of the gradient along a line provides a measure of the strength of the boundary represented by that line. On the other hand, the average gradient within a region (described in section 3.3.1.1) indicates the non-uniformity of that region.

(B) Location and Direction Features

The centroid and minimum rectangle coordinates provide the approximate location of a line. In addition, other important features include the coordinates of the starting and ending pixels. Each line has a specific direction that can be represented by an arrow along its length, as shown in figure (3.12). The average gradient direction across the line determines which of the two possible arrow directions

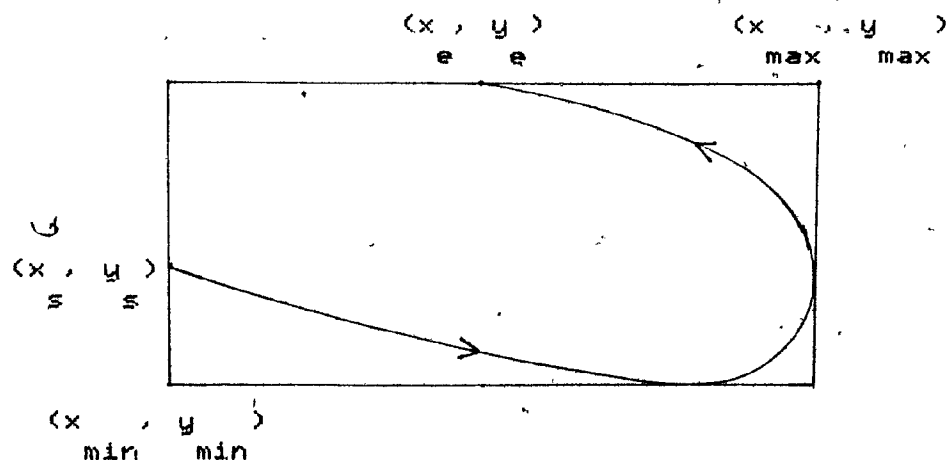


Figure (3.12) Location and direction of a line.

must be selected. Initially, the chain code of the line is computed by tracing the pixels in the map starting at either of the two terminal pixels. Thus, the direction of the arrow is arbitrarily chosen, and the result is stored in the line chain-code relation as follows:

$$\text{Line No} \rightarrow \begin{matrix} x & y \\ s & s \end{matrix}, \text{Length, Chain code} \quad (3.14)$$

At each pixel i , the element of the chain code c_i indicates the direction at which the next pixel on the line exists. This is compared to the gradient orientation o_i at that pixel in order to determine the correct line direction. If the modulus of the difference is less than two, then both directions are consistent, and vice versa. On a global basis, the following decision function is computed for line L_j :

$$d_j = \sum_{i \in L_j} |c_i - c_{i+1}| / |L_j| \quad (3.15)$$

where the vertical bars represent vector differences, according to figure (3.4a). For example, the difference between 7 and 0 is 1, 7 and 1 is 2, 6 and 0 is 2, etc. Equation (3.15) indicates the global consistency of the chain code direction and the gradient orientation on the line. If the value of d_j is greater than two, then the chain code representation is in error and must be reversed. The ending point becomes the starting point, the arrow along the line is rotated, and the chain code is recomputed and stored. This method provides a standard for representing the direction of lines. According to the definition of gradient orientation given in section 3.2.2, while travelling in the direction of the arrow, the area to the left of any line will always be brighter than the area to its right. The convention introduced here is important when comparing different line segments, as well as in computing consistent spatial relations among lines.

The starting and ending directions of a line, d_s and d_e , are taken to be the average chain code directions along the initial and final quarters of line length, respectively. The two features play important roles in computing spatial relations and in extending and joining lines.

The type of a line is a multi-valued feature that can assume one of the following symbolic entries:

(i) A line is a LOOP if it closes back on itself, i.e., the starting and ending points are eight-connected neighbors.

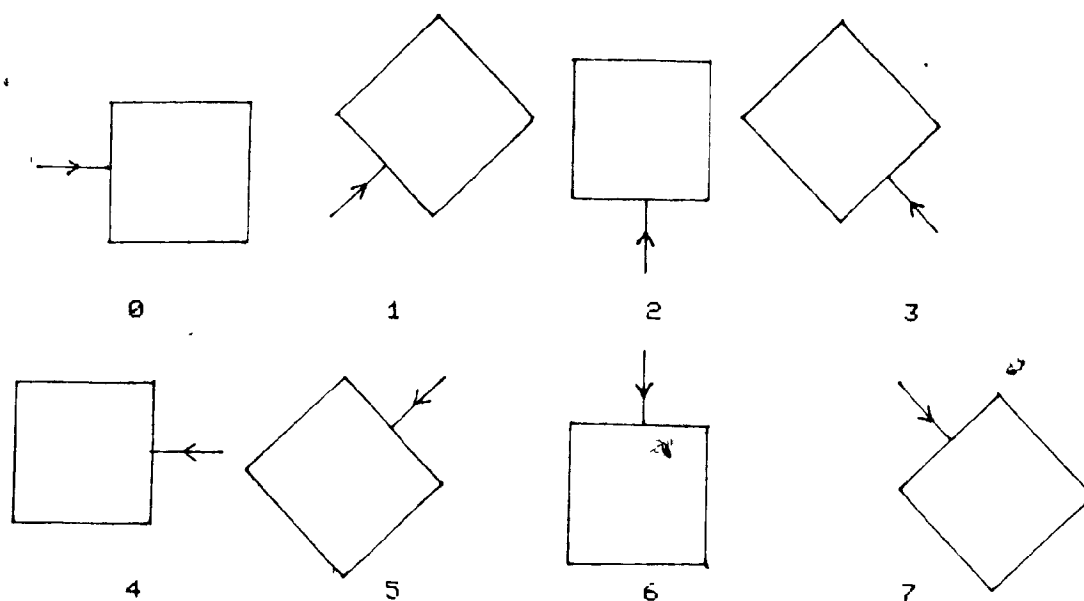
(ii) A line is START CONNECTED if its starting point has at least one eight-connected neighbor that belongs to another line.

(iii) A line is END CONNECTED if its ending point has at least one eight-connected neighbor that belongs to another line.

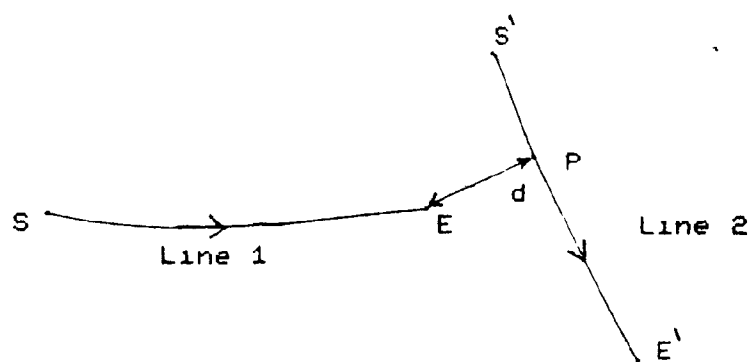
(iv) A CLOSED line has both terminal points with eight-connected neighbors that belong to other lines.

(v) Finally, an OPEN line has both terminal points open, i.e., all eight-connected neighbors that do not belong to the same line carry the null label.

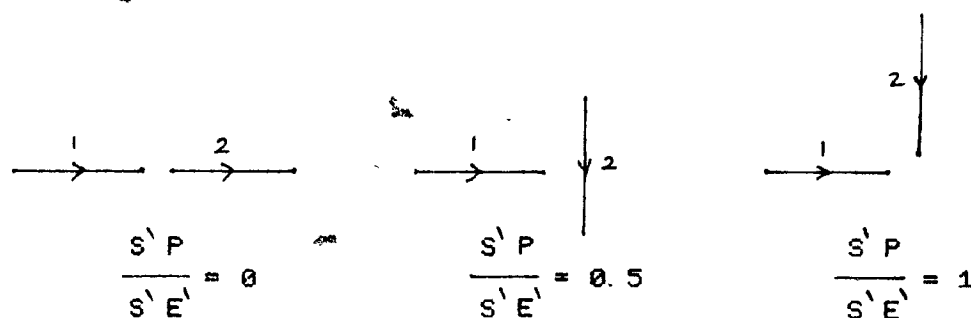
Line type provides an example of a feature that saves time during rule matching by eliminating rules that do not apply to a particular line. For example, there is no need to match on a CLOSED line rules that extend a line towards others. In this case, a condition is added at the beginning of these rules to limit matching to lines that are NOT CLOSED.



(a)



(b)



(c)

Figure (3 13) The spatial relationship IN FRONT
(a) Search areas as a function of line ending direction
(b) The relation parameters.
(c) The significance of point P

that the distance EP is a minimum. Then, the spatial relation is described by the distance EP and the position of point P relative to line 2 (the order of the point along the line chain code). Relation IN-FRONT is stored in the STM in the following manner

Line 1, Line 2 ---> Distance(EP), Position of P (3 17)

The position of point P on the second line reflects certain topological aspects of the relationship, as shown in figure (3 13c). The value stored is equal to a fraction of the total length of line 2 (0 if P is the starting point and 1 if it is the ending point). This information is used in inferences about extending and joining lines, as will be described in chapter 5.

The BEHIND relationship is the exact opposite of IN-FRONT. The search areas are established at the starting point of the line and depend on its starting direction and its length, as shown in figure (3 14a). The relation is described by the distance from the starting point S of line 1 to the nearest point P on line 2, as well as the position of P relative to line 2 (figure (3 14b)). Different values of P reflect different spatial configurations as the examples in figure (3 14c) illustrate. Relation BEHIND is stored in the STM as follows

Line 1, Line 2 ---> Distance (SP), Position of P. (3 18)

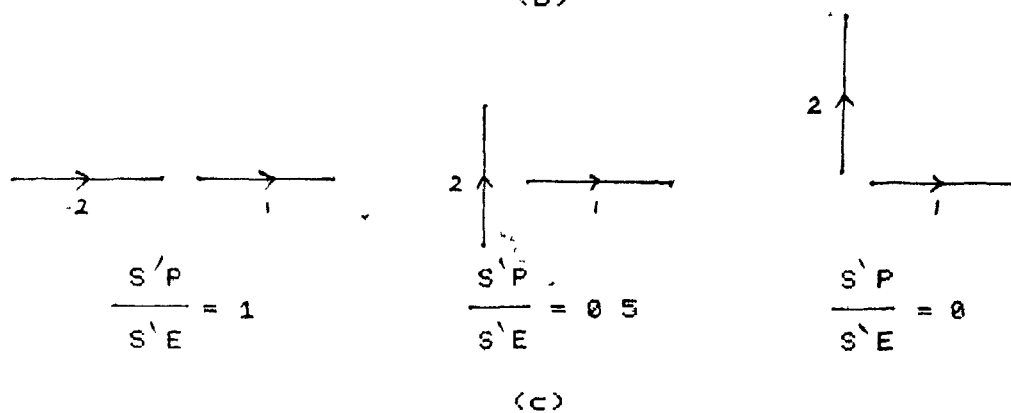
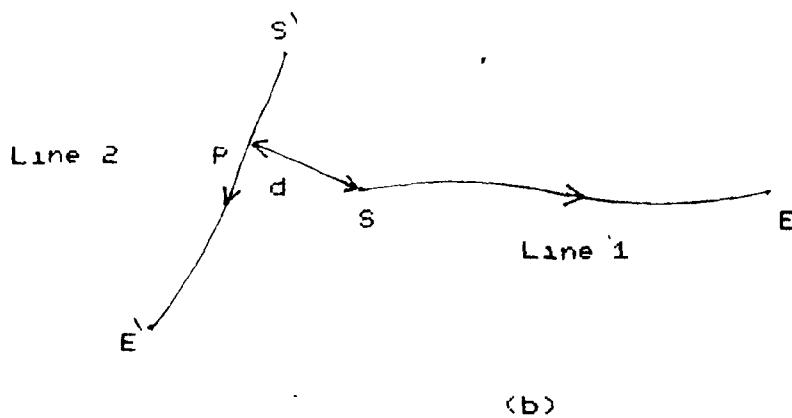
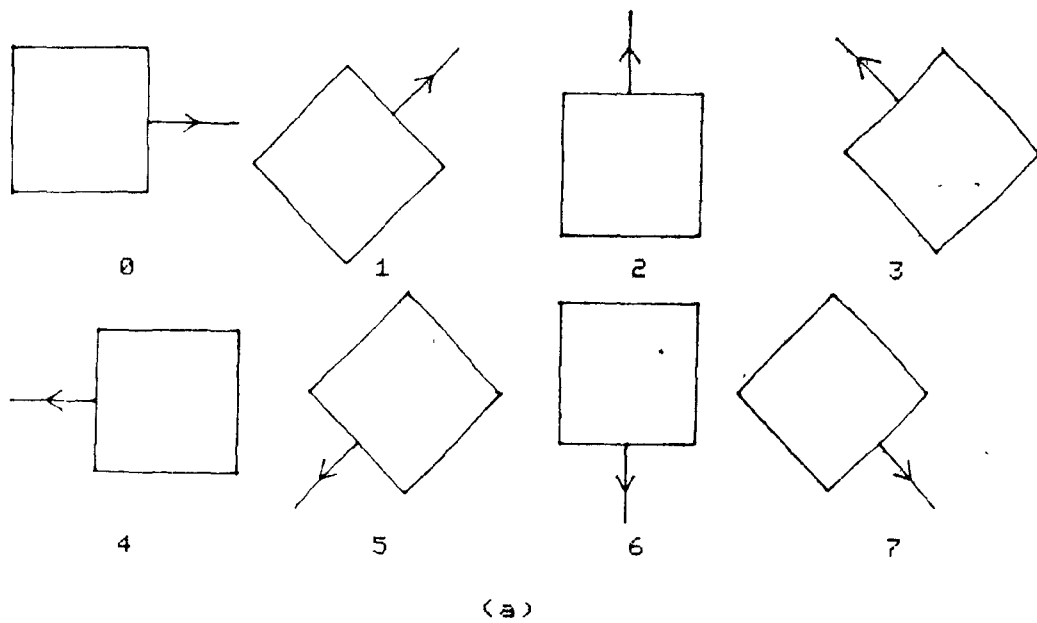
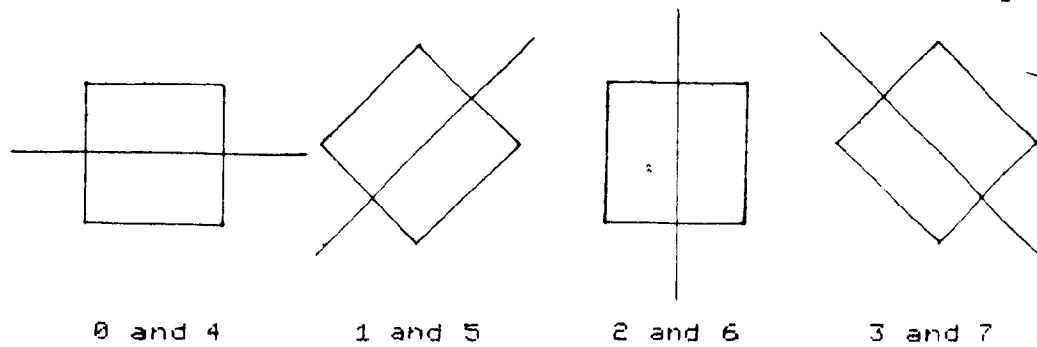
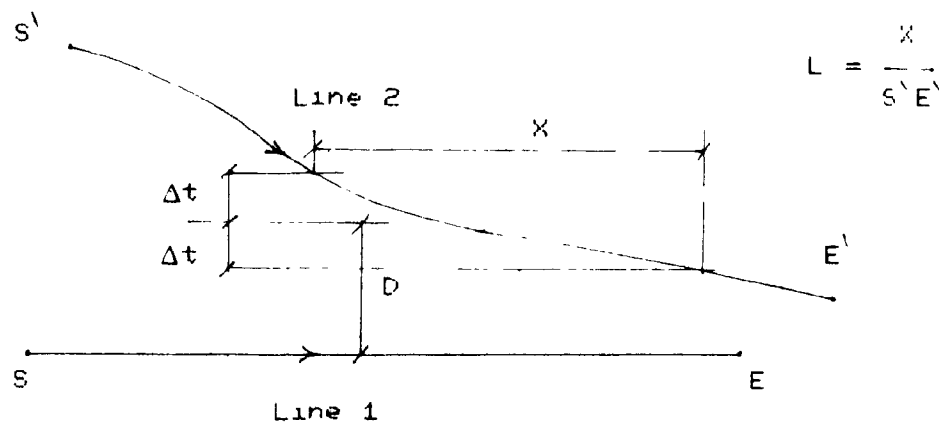


Figure (3 14) The spatial relationship BEHIND
 (a) Search areas as a function of starting direction
 (b) The relation parameters.
 (c) Different spatial configurations.



(a)



(b)

Figure (3 15) The PARALLEL relationship
(a) Search areas as a function of the average line direction
(b) The relation parameters

The third spatial relationship indicates the extent (if any) by which two lines are parallel. Again, search areas are used to detect possible candidates. These are positioned at the mid-point of the line, as shown in figure (3 15a). Whenever another line (line 2) crosses the rectangular area, the length of the segment that is parallel to the original line (line 1) is computed. This ~~is~~ accomplished by estimating the number of points on line 2 that are within a fixed tolerance Δt from the average distance between the two lines. Figure (3 15b) explains how the relationship is

computed, and introduces two parameters that describe the spatial configuration: the average distance between the two lines D , and the fraction of parallel length L on line 2. Thus, the PARALLEL relation is given by

Line 1, Line 2 $\rightarrow$ Distance D , Length L (3 19)

Table (3 3) summarizes the features stored for a line. In this case, the one at the boundary between grass and the car trunk in figure (3 11)

Information on line 113

Line type	CLOSED		
Average color features (RGB)	25 30	35 74	30 51
Variance of color features	9 64	4 54	6 51
Average gradient	18 88		
Gradient variance	11 70		
Average gradient direction	2		
Length	43		
Bounding rectangle	(175, 78) and (215, 85)		
Centroid coordinates	(195, 83)		
Starting point coordinates	(175, 85)		
Ending point coordinates	(215, 78)		
Starting direction	0		
Ending direction	7		

Lines IN FRONT (4) are

109 $\rightarrow$ 1 0 68 $\rightarrow$ 4 0 105 $\rightarrow$ 4 0 97 $\rightarrow$ 8 0

Lines BEHIND (4) are

141 $\rightarrow$ 1 0 131 $\rightarrow$ 3 0 140 $\rightarrow$ 5 0 85 $\rightarrow$ 7 0

PARALLEL Lines (1) are

73 $\rightarrow$ 7 0

Table (3 3) Example of line features

3.3.2.3. Line and Region Relations

The spatial relationships between regions and lines are classified into two classes. The LINE $\rightarrow$ REGION relations associate a particular line with the regions that surround it, while the REGION $\rightarrow$ LINE relations affiliate each region with all the lines that intersect it.

The first class contains the two relations that describe the configuration of regions to the LEFT OF and to the RIGHT OF a line, respectively. These regions are detected by examining the pixels on both sides of the line. Figure (3.16a) illustrates how the neighboring pixels are selected. Pixels to the LEFT (RIGHT) are obtained by a 90° clockwise (anticlockwise) rotation of the line chain code. Regions whose labels are attached to these pixels are considered to be to the LEFT OF (RIGHT OF) the line. The degree of the relationship is expressed by the number of pixels that belong to each region and are to the left (right) of the line, relative to the total line length. For example, in figure (3.16b), region A is to the LEFT OF line 1 by a value of 0.5, if half the pixels to the left of the line bear the region label. The two LINE $\rightarrow$ REGION relations are expressed as follows:

Line $\rightarrow$ Region Left, Value Left, (3.20)

Line $\rightarrow$ Region Right, Value Right (3.21)

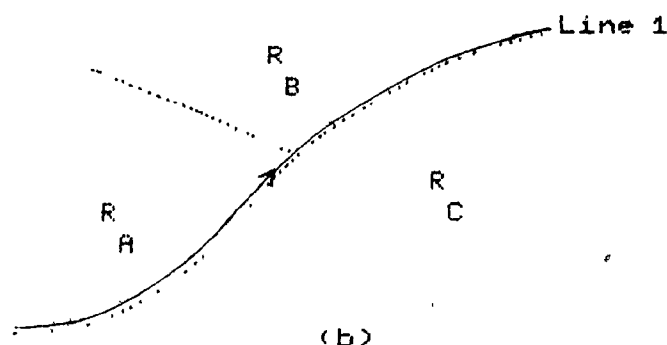
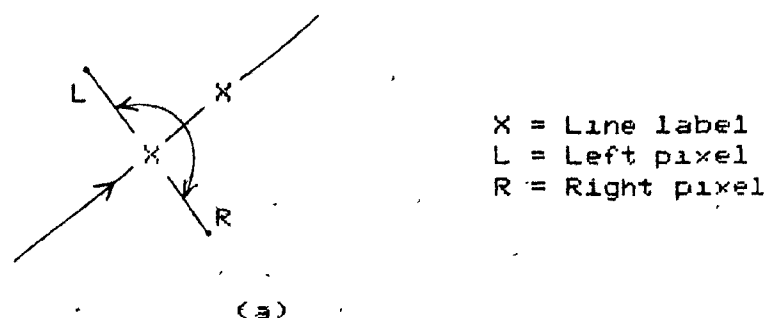


Figure (3 16) LEFT OF and RIGHT OF relationship between lines and regions

- (a) Finding the pixels to the left and right
- (b) Regions A and B are 'to the left of' line 1 with values of 0.5, and region C is to its right with a value of 1.

The REGION ---> LINE relation is constructed by finding the lines that INTERSECT each region. The fraction of the line length I that overlaps the region is used to characterize the relationship, as shown in figure (3 17). The INTERSECT relation is stored in the STM as follows:

Region ---> Line, Intersect Value (I) (3 22)

Since a line intersecting a region implies that the region must be to the left and/or to the right of the line, there is redundancy in the information stored in both relations. However, the extra storage requirements are

compensated for by the savings in computation time brought about by making the information readily available. In addition to the information in table (3 2), the spatial relationships between the grass region and the lines that intersect it are shown in table (3 4). For the line described in table (3 3), there is only one region at each of its sides with left and right values equal to 1.

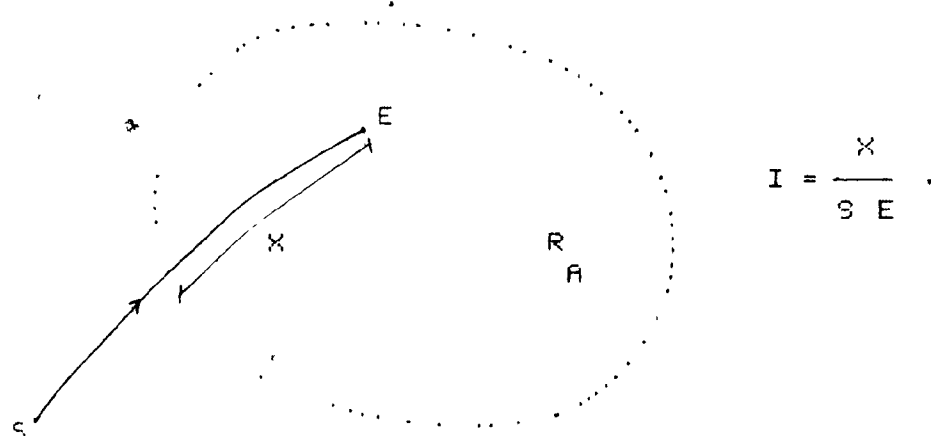


Figure (3 17) The INTERSECT relationship between regions and lines

Lines intersecting region 26 are (43)

99 --> 1 00	109 --> 1 00	113 --> 1 00	122 --> 1 00
132 --> 1 00	143 --> 1 00	145 --> 1 00	147 --> 1 00
148 --> 1 00	156 --> 1 00	165 --> 1 00	212 --> 1 00
134 --> 0 95	154 --> 0 93	178 --> 0 92	157 --> 0 91
106 --> 0 89	123 --> 0 89	124 --> 0 89	168 --> 0 88
100 --> 0 86	149 --> 0 86	97 --> 0 85	114 --> 0 78
199 --> 0 77	153 --> 0 74	177 --> 0 72	141 --> 0 72
94 --> 0 60	197 --> 0 53	172 --> 0 52	105 --> 0 50
222 --> 0 50	116 --> 0 45	164 --> 0 33	126 --> 0 25
50 --> 0 24	175 --> 0 21	167 --> 0 20	125 --> 0 14
86 --> 0 13	144 --> 0 13	198 --> 0 13	

Table (3 4) List of lines intersecting a region

3.3.3. Focus of Attention Areas

In addition to maintaining regions and lines separately, they are grouped together into areas according to specific criteria. A description of these areas is also stored in the STM for use by the processing modules. The main reasons for creating this extra data entry can be summarized as follows:

(1) The need for a spatial partition of the image into independent areas that can be treated by the processing modules in accordance with their individual characteristics. Measurements of performance are computed for each area, and the processing strategy is adjusted to accommodate the particular requirements reflected by these parameters.

(2) Areas carry structural information that is not present in regions or lines alone. The additional data can be used by the high level process in assigning object interpretations to parts of the image. A prime example is in the detection of textured areas within the image. For outdoor scenes, the properties of such areas will suggest whether they represent the crown of a tree or the pattern on a brick wall.

(3) The segregation of regions and lines into spatially independent groups allows for the possibility of parallel processing of the image. This alleviates the worry about synchronization since different processes can work on separate areas without interference.

Areas are generated based on the region and line configurations stored in the STM. Chapter 4 contains the rules that are used to create and maintain different types of focus of attention areas. For each new area, a number of properties are computed and stored. Some are features that describe the position and topological structure, others are used in determining the processing strategy within an area, as will be discussed in detail in chapter 6. Before defining the stored attributes for areas in the STM, we will first introduce the three types generated by the system.

3.3.3.1. Types of Areas

The model stored in the LTM contains rules that examine the region and line properties described in sections 3.3.1 and 3.3.2 and consequently generate one of the following types of areas:

(1) Smooth Areas

These are characterized by large regions that exhibit little variance in their measured features. They are easily detected by simple region analysis techniques, because of their size, as well as the uniformity of their features. Therefore, they are usually composed of a small number of regions (or, in some cases, a single region). For example, in natural scenes, areas that correspond to the sky are usually smooth. Such areas are also reflected in the line map where, in normal situations, they will not contain any

lines. Therefore, the detection of a line (especially a short one) will in most cases indicate an error in the line partition.

(2) Textured Areas

A cluster of a large number of adjacent regions of relatively small size and different average features represents strong evidence of texture. This is because a textured area consists of basic elements of different intensity that alternate in some periodic manner. Each element is uniform within itself, but bears a distinct dissimilarity to its adjacent neighbors (a simple example is the checkerboard pattern). Therefore, the regions that are created based on uniformity analysis will correspond to the elementary units of texture (the squares in the checkerboard). The spatial configuration of these regions and the periodicity of their features define the area as a whole (the entire board).

An example of texture in natural scenes is the crown of a tree, where the leaves are interlaced with a uniform background such as the sky. Region merging and splitting methods will result in a large number of regions, with each corresponding to a single leaf or to a patch of sky that shows through. Such data configurations in the region map are considered candidates for constituting textured areas. Once detected, an area can be examined closely to determine whether indeed it belongs to a particular texture pattern. A comparison between this approach to texture analysis and

previous methods will be given in chapter 6

(3) Bounded Areas

Long lines indicate the presence of a prominent boundary within the image. The fact that an edge exists consistently over a considerable number of pixels contributes to our confidence in its validity as a major separator within the image. In certain situations, the long boundaries can be used to decompose the image into principal areas of attention. Alternatively, separate areas are formed when long lines close (or almost close) on themselves. An area enclosed (or bounded) by such a loop may represent a particular entity or phenomenon in the image. For example, lines that fall within a textured area are short and numerous because they correspond to boundaries between the individual elements of texture. By contrast, a boundary that delimits two different textures is much longer.

3.3.3.2. Descriptive Features

As with regions and lines, the position, structure, and internal properties of areas are computed and stored in the STM. In addition, a number of features describe the state of the segmentation within the area. These include the number of regions and lines that fall within the area boundaries. The size of the area is equal to the sum of the sizes of its constituent regions. The total length of the lines within the area is also stored. Finally, measures for region

uniformity, the contrast between all adjacent regions, the contrast across all lines, and the connectivity of the lines constitute a performance vector for the area. These measures will be described in detail in section 6.3. The description relation for areas is given by

```
Area No ---> Flag, (xmin, ymin), (xmax, ymax),
               No. of Regions, No. of Lines,
               Area Size, Total Line Lengths,
               Region Number, Line Number,
               Variance L,           , Variance M,
               Uniformity L,         , Uniformity M,
               Peg Contrast L,       , Reg. Contrast M,
               Line Contrast L,     , Line Contrast M,
               Line Connectivity      (3 23)
```

The flag indicates whether the area is active (Flag = 0), inactive (Flag = -1), or if it has been deleted (Flag = -2). Areas are rendered active or inactive by the AREA ANALYSER according to the state of the segmentation so that they can be brought in and out of the analysis, respectively.

A list of the regions and another for the lines that belong to each area are also maintained. These are stored in the form of a chain of links between successive pairs in the lists. The two STM relations involved are termed AREA-REGIONS and AREA-LINES, and have the form

```
Region A ---> Region B , (3 24)
and Line A -> Line B , (3 25)
```

where A and B indicate a pair of regions or lines that belong

to the same area. The method of storage is best described by the example in figure (3.18a). The area contains regions 3, 4, 7, 9, and 12. A pointer to region 3 is stored in the description relation. Entries that link the other regions together are included in the AREA-REGIONS relation, as shown in figure (3.18b).

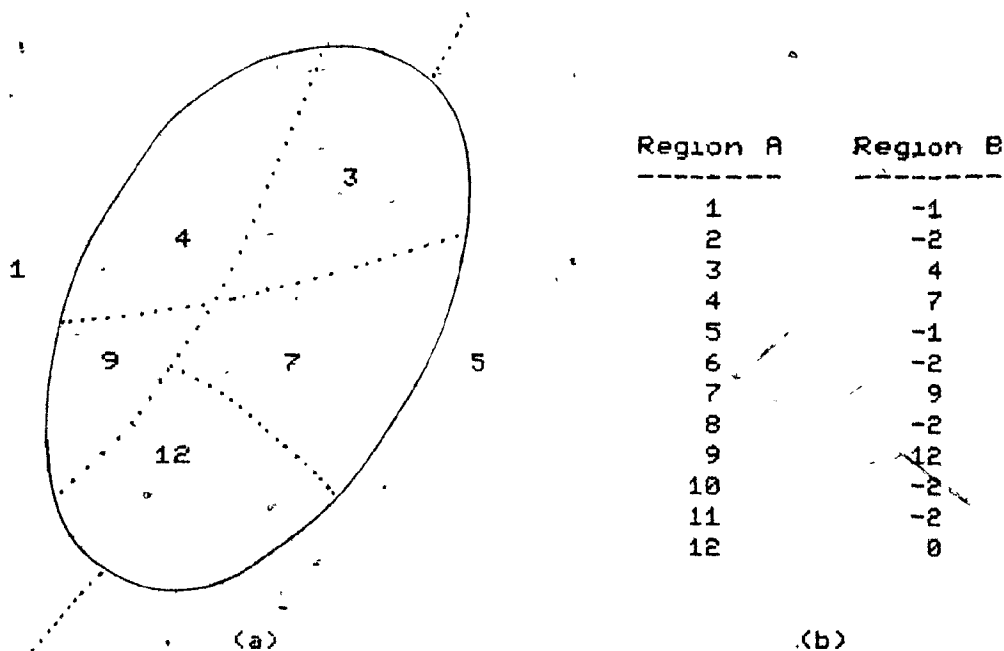


Figure (3.18) Example of storing a list of regions for one area.

Accessing all the regions that belong to the area is reduced to the simple task of following the list starting at the label obtained from the area description relation. Since each region can only belong to one area (i.e., areas do not overlap), all areas can use the same relation to store their region labels. In general, all the region labels are included under the REGION A column of the relation in equation (3.24). The REGION B column can contain one of the

following (i) another region label that belongs to the same area as REGION A, (ii) zero if REGION A is the last region in an area's list, (iii) -1 if REGION A does not belong to any area, (iv) -2 if REGION A is not in the region map

The above arrangement is quite useful in finding whether a region already falls within an area, a fact needed by the area generation process. The same method is used for storing the line labels for each area.

This section described the various attributes of regions, lines, and areas that are stored in the STM. Some of these are used directly in the matching process in order to determine whether the conditions of rules are satisfied. Others help in executing the actions of rules by providing pre-computed information, which would delay processing considerably if they were to be calculated only when needed. Chapter 5 will discuss the rule matching process in detail and will enumerate, in addition to the features described above, others that exist in rule conditions and are evaluated each time they appear. It should be noted that the selection of the features to be stored prior to rule matching and those that are computed during processing depends on two factors: the frequency of usage, and the amount of computation required.

3.4. THE OUTPUT

Starting with an initial configuration of regions and lines, the rules are used to generate areas of attention and consequently modify the regions, lines, and areas until a final result is achieved. Therefore, the output of this rule-based segmentation process includes a set of regions which partition the image into uniform sub-parts, a second partition of lines, and a grouping of some of the regions and lines into areas. Although the regions and lines are maintained and updated separately, the criteria embodied in the rules will use the information from both sources to ensure a consistent representation in the two partitions. As the processing progresses, each map will approach its final status, and at the same time, the lines will gradually conform with the boundaries of the regions.

A conflict may arise in the final output, if some of the remaining lines have open ends, and are not part of a region boundary. Because the boundaries of regions are closed, it is not possible to partition a region along an open line. In most cases, the rules will attempt to delete such lines, based on the surrounding region information. However, as discussed in section 3.3.2, open lines may in fact represent a physical instance of a thin object in the scene. Two solutions are possible: the inclusion of these lines in the final output, or the creation of thin regions that have a double boundary at the same pixel locations. In either case, the source of this problem lies in the limitations presented

by the finite resolution of the measured image

As an integral part of an image understanding system, the output of the low level segmentation process is delivered to higher stages of analysis. Current interpretation systems use a region partition to describe the image [Levine and Shaheen, 1981]. The data that describe the final configuration of lines provides an additional and rich source of information. For example, the strength of the boundaries presents a useful clue to a semantic interpretation process [Yakimovsky and Feldman, 1973]. Moreover, the areas of attention introduced in this system, carry supplementary conclusions that can prove useful in identifying parts of the image, as described in the previous section.

Besides providing an input to the next stages of analysis, we are interested in the output segmentation as being a direct product of the rules in the model. In order to test and order our knowledge base, we must have the ability to quantitatively evaluate the output. Because a low level segmentation is not unique, its quality can only be judged relative to human expectations. The following measure of segmentation has been designed to simulate such a task.

3.4.1. Measure of Segmentation

The measure described here compares two image partitions by computing a two-dimensional distance estimate. If one of the partitions, taken as a reference, is produced manually,

then the measure would indicate how far the other partition is from that reference. A larger distance implies a greater error in the tested output relative to the human generated one. Therefore, this comparison measure can be used to evaluate the performance of a particular algorithm, as well as to rank different outputs.

The two-dimensional distance referred to above has a component proportional to the amount by which the regions in a tested segmentation partition those in the reference. The other component signifies the amount by which the tested regions overlap the reference regions. Therefore, the two components indicate the degree of under-merging and over-merging in the test segmentation with respect to the reference.

Consider a reference segmentation output containing the set of N regions $[R_1, R_2, \dots, R_N]$ having areas $[A_1, A_2, \dots, A_N]$ respectively, and a test segmentation output containing the set of M regions $[r_1, r_2, \dots, r_M]$ having areas $[a_1, a_2, \dots, a_M]$. The under-merging error measure is then given by

$$UM = \frac{\sum_{1 \leq j \leq M} (A_k - \cap[r_j, R_k]) \cap[r_j, R_k]}{A_k}, \quad (3.26)$$

where, $\cap[r_j, R_k]$ is the area of the image shared between r_j and R_k ; i.e., the number of pixels that have label r_j in the

test output and label R_K in the reference, and R_K is the reference region such that

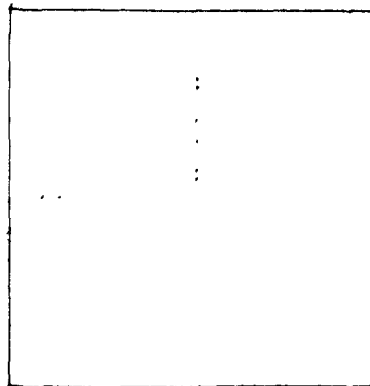
$$\bigcap_{j \in K} [r_j, R] = \max_{1 \leq i \leq N} \left(\bigcap_{j \in K} [r_j, R_i] \right) \quad (3.27)$$

In other words, for each tested region, the reference region that has maximum overlap is selected. The difference between the value of the overlap and the area of the reference region is proportional to the under-merging error. The contribution to the measure is computed by scaling this difference by the ratio of the overlap to the reference region area. The simple examples in figure (3.19) show a single reference region that is partitioned by several test regions (i.e., with no over-merging). The value of the under-merging component is indicated under each case, where A is the area of the reference region.

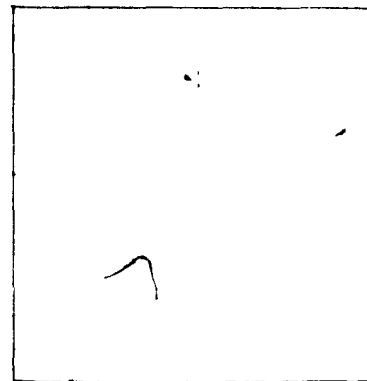
The over-merging error measure is given by

$$OM = \sum_{1 \leq j \leq M} \left(a_j - \bigcap_{j \in K} [r_j, R] \right) \quad (3.28)$$

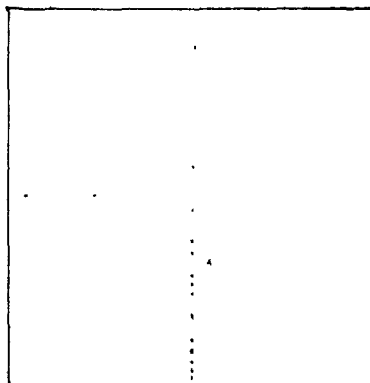
The balance of the area of a region in the test output that is not covered by the reference region with maximum overlap is proportional to the over-merging error. Notice that if the test region is totally contained within the reference region, its contribution to the measure is nil.



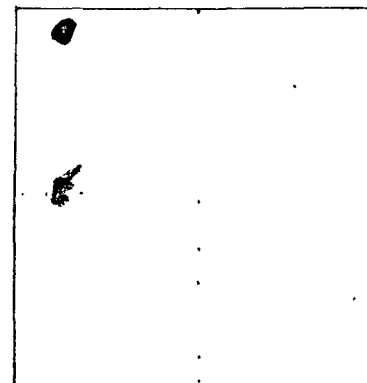
$$\begin{aligned} & (1/4)A \times 3/4 \\ + & (3/4)A \times 1/4 = (3/8)A \end{aligned}$$



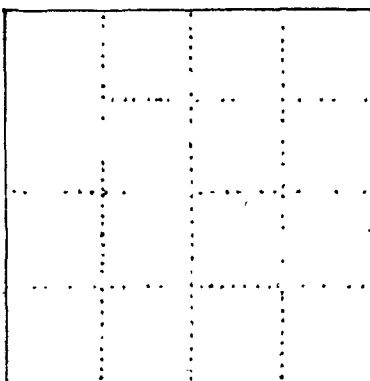
$$\begin{aligned} & (1/2)A \times 1/2 \\ + & (1/2)A \times 1/2 = (1/2)A \end{aligned}$$



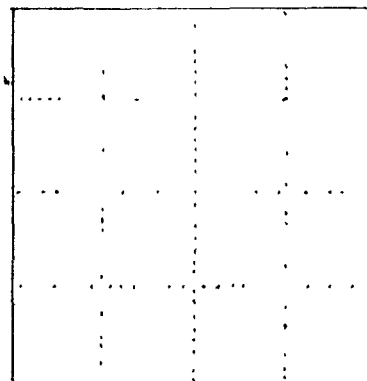
$$\begin{aligned} & (1/2)A \times 1/2 \\ + & (3/4)A \times 1/4 \\ + & (3/4)A \times 1/4 = (5/8)A \end{aligned}$$



$$4 [(3/4)A \times 1/4] = (3/4)A$$



$$\begin{aligned} & 14 [(15/16)A \times 1/16] \\ + & (14/16)A \times 2/16 = (238/256)A \end{aligned}$$



$$\begin{aligned} & 16 [(15/16)A \times 1/16] \\ & = (240/256)A \end{aligned}$$

Figure (3 19) Examples of under-merging error computation

It can be easily shown that both measures have a lower bound of zero, when the test and reference outputs are identical, and a higher bound equal to the area of the image minus one. The two measures can thus be normalized and combined to produce one of the following composite error estimates

$$M_1 = \left[\left(\frac{UM}{A_I} \right)^2 + \left(\frac{OM}{A_I} \right)^{1/2} \right], \quad (3.29)$$

$$\text{or} \quad M_2 = \left[\frac{UM}{A_I} + \frac{OM}{A_I} \right], \quad (3.30)$$

where A_I is the area of the image

In order to evaluate the line partition, an additional set of measures is required. By comparing a line map to a reference region map, two sources of errors will become apparent

- (1) The existence of a line within a reference region, i.e., a boundary was detected where none actually exists
- (2) The absence of a line at the boundary between two reference regions, i.e., an actual boundary was not detected

Therefore, a two-dimensional comparison measure with components for excess lines and missing lines will account for the above errors. The two error measures are computed by superimposing the line map on the reference region map. For each line L_j , the portion of line length n_j that falls at the boundary between two reference regions is computed (n_j is the number of line pixels that have different reference regions

on alternate sides) These values are then added for all the lines in the tested line map The length of reference boundaries NOT covered by lines is found by subtracting the above total from the lengths of all reference boundaries BL This is then normalized to obtain the missing lines error measure ML, as follows

$$ML = \frac{BL - \sum_{\text{all } L_j} n_j}{BL} \quad (3.31)$$

The balance of the length of lines that do not fall on reference boundaries is divided by the total line lengths to produce the excess lines error measure EL, as follows

$$EL = \frac{\sum_{\text{all } L_j} (l_j - n_j)}{\sum_{\text{all } L_j} l_j} \quad (3.32)$$

where l_j is the length of line L_j The two components can be combined to produce a composite measure similar to those in equations (3.29) and (3.30)

Other measures that are based on the idea of comparing two partitions have been described in the literature They method introduced by Coleman computes a statistical measure from the joint histogram of the segmented images [Coleman, 1977] In effect, this comparison measure

estimates the number of misclassified image points in one output relative to the other. The disadvantage of this method lies in the loss of spatial information in computing the histogram. This leads to cases where the measure does not seem to agree well with human observation. To correct for such errors, a more complex measure has been developed that compares the manual partition of cell images with computed ones [Yasnoff, Muir, and Bacus, 1977]. The position of misclassified pixels is taken into account, as well as their number. An estimate of the error is taken to be proportional to the sum of the distances between pixels that have been incorrectly assigned to a particular class and the nearest pixels that actually belong to that class. The method appears to perform well for the cell images which have well defined classifications for the pixels (e.g., nucleus, cytoplasm, red cells, and background). In the absence of this contextual information, the measure is not computable because the classes are not defined. In contrast to the above, the procedure described above provides a general purpose low level comparison measure that can be applied to any class of images.

A different type of performance measurement has been developed in conjunction with edge-based segmentation techniques. Fram and Deutch [Fram and Deutch, 1975] computed two measures that evaluate the output of edge detectors. The first estimated the fraction of correctly detected edge points in a synthesized image of an edge in the presence of

noise. The second computed the fraction of the actual edge that was covered by detected pixels. Their results show the deterioration in the performance of edge detectors with noise. They were also able to rank several mask operators and demonstrate the consistent superiority of some schemes over others.

Abdou and Pratt [Abdou and Pratt, 1979] employ an analytical approach to study the sensitivity of detectors to edge orientation. In addition, they used a figure of merit for edge quality [Pratt, 1978] that is based on the displacement of the detected edge point from the ideal edge position. As with the previous method, this requires the knowledge of the correct location of the edge in the synthesized test images. The results show an ordering of the edge detectors that is quite similar to that obtained by Fram and Deutch.

Bryant and Bouldin [Bryant and Bouldin, 1979] proposed a measure that is based on the correlation of the output of edge detectors with an "ideal" output, presumably produced by hand. In an effort to eliminate the need for the knowledge of the correct output, they described a second measure that computes a consensus of several edge detectors on the presence or absence of edge points. Each detector is then evaluated on the basis of how well it conforms to (or differs from) that consensus. The results are sensitive to the subset of operators used to compute the consensus. If a consistent error is part of the consensus, the error will not

be detected

Local edge coherence was used as the basis for the evaluation given by Kitchen and Rosenfeld [Kitchen and Rosenfeld, 1981]. The method evaluates the consistency of the gradient directions at pixels in the vicinity of each edge point. Detected edges are rewarded for continuity and thinness. The method suffers from a presumption that the detected edges are in their proper position. Therefore, it does not penalize consistent errors in mislocating edge points. The authors note that this renders the method valid only as a supplement to existing measures. Test results on computer generated images of edges with superimposed noise generally agree with previous findings with regard to rating various edge mask operators.

The approaches mentioned so far describe comparison measures that require a reference segmentation in order to evaluate performance. However, in automatic processing, the human element is usually not available, rendering these measures useless for real-time utilization. Consensus measurement and edge coherence methods realize the importance of eliminating such a dependency. Nevertheless, they do not provide general purpose measures that can function without the knowledge of the correct segmentation, or some other precomputed reference (as in the consensus case). What is required is a comparison between the measurements in the input image and the computed partition (the intermediate and final output). Figure (3.20) illustrates the difference

between a real-time measure, and one that is used for testing and comparison

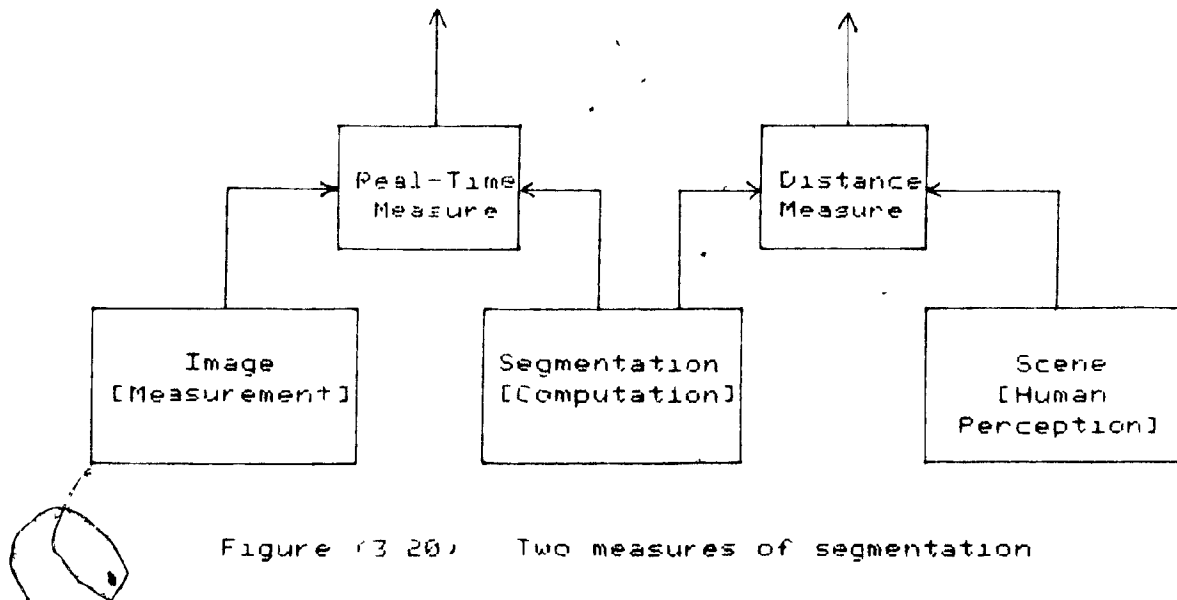


Figure (3.20) Two measures of segmentation

In chapter 6, we will introduce a set of optimization criteria that are involved in measuring the performance of image segmentation systems. These are intended as general-purpose real-time parameters that can be used (all or the appropriate subset) in evaluating, as well as guiding, the processing of any partitioning algorithm. They apply to region-based, line-based, and texture-based approaches to segmentation. In particular, these measures are designed to provide the means for setting the processing strategy of the rule-based image segmentation system presented in this thesis.

3.5. SYSTEM INFORMATION AND STATISTICS

In addition to image and segmentation data, information about the processing history and various control parameters is accumulated in the STM. These data are used by the meta-rules and the focus of attention rules that execute the system control strategy, as will be described in the following chapters. Furthermore, statistics about the rules are used to establish a dynamic learning procedure that orders the rules based on past performances.

The following sections classify the system parameters into two main categories: processing information and rule statistics.

3.5.1. Processing Information

A rule cycle is defined as the period of time that elapses between successive invocations of any of the analysis modules. During this period, the set of knowledge rules belonging to one module is matched against the data in the order specified by their priorities, until one rule matches or all rules are tested. Therefore, each cycle defines a particular processing module (REGIONS, LINES, or AREAS), as well as a rule action (e.g., merge, split, extend, delete). The latter may be the NULL action in the case where none of the knowledge rules match. A record of processing to date is stored in the SYSTEM STATUS relation as follows:

Cycle No ---> Process, Action, No of actions,
 Area No , Region No , Line No ,
 Under Merging Measure [optional],
 Over Merging Measure [optional],
 Excess Lines Measure [optional],
 Missing Lines Measure [optional] (3 33)

The number of actions is the total to the current cycle, and is increased by one if a knowledge rule fires and modifies the data. The labels of the current area, region, and line are also stored. They change in cycles where a focus of attention rule fires to obtain new entries. If a reference segmentation is available, the four error measures introduced in the previous section are updated at the end of each cycle. This provides a continuous profile of progress that will be used in chapter 7 to analyse different knowledge sources and control strategies. Chapter 5 discusses how the information gained for the previous cycles is used by the control rules to select the next process within the framework of a given strategy.

3.5.2. Rule Statistics and Priority

When a knowledge rule fires and modifies the segmentation (e.g., by merging two regions), the performance measurements of the area (that contains the two regions) will change. These incremental changes in performance are gathered for all actions produced by the same rule. Their sum provides a measure of the effectiveness of the rule in enhancing performance. For example, a merging rule will tend to improve the contrast between regions in an area, but it

may also decrease the uniformity of individual regions. If the sum of the increments over a period of time is positive, the rule is considered beneficial, and vice versa.

Another factor that influences the choice of rules is the frequency with which they fire. The hit ratio of a rule is defined as the number of times all its conditions match the data, divided by the number of times the rule was tested. The rule statistics relation stores the information about both the activity and effectiveness of all the rules, and is expressed as

$$\text{Rule No.} \rightarrow \text{Hit ratio, } \Delta P_1, \Delta P_2, \dots, \Delta P_M \quad (3.34)$$

Rules that perform the same action (e.g., deleting lines) are considered to fall within the same category. In chapter 6 we will define the priorities among different rule categories as part of the system strategy. However, the ordering of the rules within each category is obtained through learning. Rules are matched in the order in which they appear in the rule priority relation, which has the form

$$\text{Rule category, Effectiveness, Activity} \rightarrow \text{Rule No.} \quad (3.35)$$

Rule effectiveness is a summation over all increments in performance measures caused by the rule. The activity of a rule is equal to its hit ratio. Both quantities are normalized and quantized into just five categories using a scaling method that will be described in chapter 5. The

relation is sorted according to its three key domains. The first isolates the rules by their categories, the second arranges the more effective rules first within each category, and finally the third places the less active rules ahead for those with equal effectiveness. The relation is dynamically updated using the information stored in the rule statistics relation of equation (3.34).

Long term statistics about the rules are also stored in a relation similar to (3.34), except that it is part of the LTM. The effectiveness and activity measurements described above are accumulated over several program runs involving different images, thus providing a global means of evaluating the rules. The information in the LTM relation is used to initially order the rules when processing a new image. With the exception of the last relation, this chapter introduced the image specific data stored in the STM. The next chapter will discuss how the data is modified by the processing modules, while chapter 5 will describe the image independent data stored in the LTM.

CHAPTER 4

THE PROCESSING MODULES

The processing modules shown in figure (1.1) are responsible for carrying out the rule-based analysis. Their function is to apply the production rules stored in the LTM to the STM data. Starting with initial image partitions, the different modules will attempt to place the knowledge represented by the rules in conformity with the measurements in the image. The mechanism is the same for all processing modules and is composed of test-and-modify cycles. Each module matches the conditions of a rule to a particular data item, and if the match is successful, the changes specified by that rule are applied to the data.

The six modules are designed as modular units that perform specific functions. The INITIALIZER produces region and line partitions for the remaining modules to operate on. The AREA ANALYZER generates and maintains focus of attention areas, while the REGION and LINE ANALYZERS improve the

initial partitions and ultimately produce a final output segmentation. The FOCUS OF ATTENTION module selects the areas, regions, and lines that are tested at any point during processing. Finally, the SUPERVISOR coordinates the operation of all the processing modules, including the all important role of commencing and ending the analysis.

While each module has its own set of rules, they all use the same interpreter in parsing the conditions to determine the required tests. The features described in the previous chapter are available for any of the rules to examine. It is the action part that identifies the module to which a rule belongs. For example, the actions of the REGION ANALYZER rules include those that split a region into many and others that merge two regions together. Before discussing in chapter 5 WHAT each module can accomplish by applying its rules, the following sections describe HOW the data is examined and modified.

4.1. THE INITIALIZER

When a new image is read into the STM, the INITIALIZER is brought into action to provide a primary set of segmentation data. This process is necessary for the proper functioning of all the other processing modules. The REGION and LINE ANALYZERS can only match their rules on existing configurations of regions and lines. Similarly, the AREA ANALYZER will generate areas of attention based on such partitions. Therefore, a SUPERVISOR rule must insure that

the INITIALIZER is the first module to handle the data. Following is a discussion of how the data are prepared for the ensuing processing.

4.1.1. Initializing Regions

At the beginning of the analysis a complete region map must be generated in which a region label is assigned to every pixel in the image. Because both merging and splitting are used in the analysis, the initial region partition can be arbitrarily chosen. On the other hand, the INITIALIZER may employ a simple and fast grouping technique to produce a manageable number of regions. One example of the latter is to assign the same region label to all adjacent pixels that have the same grey level intensities (or color), a procedure that makes use of one elementary rule and requires a single pass over the image. Arbitrary segmentations are produced by dividing the image into squares of size $n \times n$, where n varies between 1 (when each pixel is considered as a new region) and the size of the image (which becomes a single large region), as shown in figure (4.1). The effects of the initial region configuration on the efficiency of processing and on the final outcome will be investigated as part of the experimental results presented in chapter 7.

Once the region map is initialized, it is used in conjunction with the input and gradient images to compute and store the region features described in section 3.3.1. The region map is scanned and the location, size, shape, and

statistical features of all regions are recorded. The chain code descriptions of the region boundaries are also extracted from the map, and are then used to identify all adjacent regions and store their adjacency values. As soon as the features are computed, the initial region partition becomes ready for the application of the knowledge rules, a task reserved for the REGION ANALYZER.

1	2	3	4	5	6	7	8	1	1	1	1	1	1	1	1	1
9	10	11	12	13	14	15	16	1	1	1	1	1	1	1	1	1
17	18	19	20	21	22	23	24	1	1	1	1	1	1	1	1	1
25	26	27	28	29	30	31	32	1	1	1	1	1	1	1	1	1
33	34	35	36	37	38	39	40	1	1	1	1	1	1	1	1	1
41	42	43	44	45	46	47	48	1	1	1	1	1	1	1	1	1
49	50	51	52	53	54	55	56	1	1	1	1	1	1	1	1	1
57	58	59	60	61	62	63	64	1	1	1	1	1	1	1	1	1

Figure (4.1) Initial region partitions
 (a) Each pixel is a region
 (b) The whole image is a single region

4.1.2. Initializing Lines

An initial configuration of lines is obtained by grouping together adjacent edge pixels. The latter are obtained by thresholding the gradient image, and therefore correspond to local discontinuities in the image array. Threshold selection has a direct effect on the initial lines. For example, a low absolute value will produce a large number of edge points. Consequently, the resulting map will contain numerous lines, some of which do not correspond to physical boundaries in the image. Alternatively, a high threshold

value leads to less edge points, and hence a line map that lacks representation of the weaker boundaries

Adaptive thresholds that depend on the gradient within a specific neighborhood can also be used [Robinson, 1977]. They tend to suppress multiple edge indications, and hence generate less lines that represent the same boundary. However, because they will detect an edge at any point that is above average, even at considerably low gradient values, adaptive thresholds are more susceptible to noise.

In chapter 7, we will experiment with the rule-based system's ability to recover from different faults in the initial line partition. Excess representation produced by low thresholds triggers rules that delete noisy lines, while missing lines that result from high thresholds are detected by rules that extend and complete the line map.

Using absolute or adaptive thresholding, a binary edge array is produced from the gradient magnitude array. The method by which the edge pixels are grouped into lines relies on the gradient direction at each pixel. The edge array is scanned until an edge pixel is encountered. The gradient direction at that pixel is then examined, and three candidate pixels are determined so that the resulting line orientation will be compatible with the overall gradient direction across the line. Figure (4.2) indicates the order in which the candidates are examined for each gradient direction. The first candidate that is also an edge point is selected as the

next addition to the line. This grouping process iterates until an end-point is reached where all candidates are non-edge pixels. The scanning process will then continue to search for the beginning of a new line, at which point the grouping process is reinstated. Scanning and grouping alternate until all the edge pixels are accounted for. A flow chart of the line generation process is shown in figure (4 3).

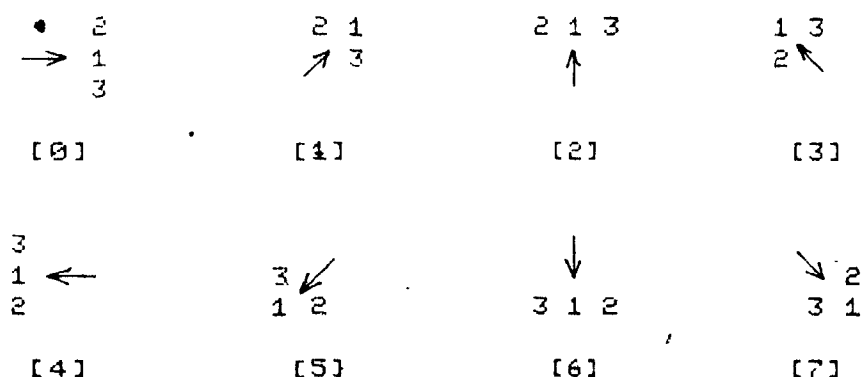


Figure (4 2) Candidate pixels for line grouping

The process described above will result in a labelled line map, as well as a chain code description of all detected lines. The attributes described in section 3 3 2 are extracted from each line by tracing along its length and recording the statistical and structural features. Search areas are then established to detect neighbors and compute their spatial relationships. Finally, the spatial relationships among initial regions and lines are computed and stored.

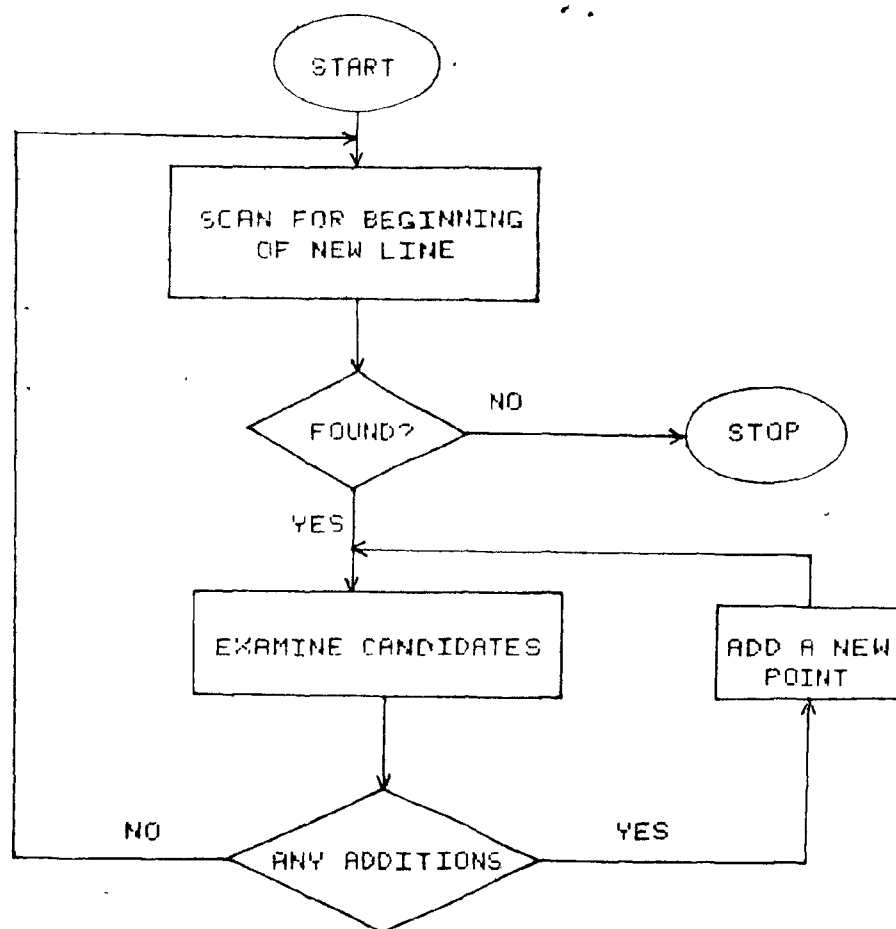


Figure (4 3) A flow chart of the line generation process

The INITIALIZER provides the primary data structure to be used in the analysis. Although its presence is necessary, it is designed to have minimal influence on the ensuing processing. The final outcome will depend mainly on the rules used and not on the initial configuration of regions and lines. This is achieved by including both merging and splitting rules, as well as rules that add or delete lines. Consequently, it is possible to study the effect of the initial partitions on both the output quality and the processing efficiency. Obviously, different initial

partitions will result in varying degrees of processing efficiency (e.g., if the initial segmentation is correct, no processing will be required). The segmentation at the onset of processing must be chosen on the basis of reducing its influence on the final outcome, as well as minimizing computations, as will be demonstrated in chapter 7.

4.2. THE REGION ANALYZER

The actual processing of regions takes place when the REGION ANALYZER matches its rules against the STM data. The initial region partition is successively modified by the rules which apply the criteria to be embodied in the final outcome. The region map will change when a rule fires in one of two ways: two regions are merged into one, or a region is split into several.

4.2.1. Merging Regions

Two regions can only be merged if they share a common boundary. The REGION ANALYZER makes use of the stored attributes of both regions to accomplish its goal. The following steps are executed when regions R_i and R_j are merged:

- (1) The new region acquires the label of the first region (R_i), while the second label (R_j) is freed for future use.

(2) Position features are used to locate pixels in the map that carry the label of the second region and change their label to that of the first. The new region boundary is traced and its chain code description is stored.

(3) The size of the new region is equal to the sum of the sizes

$$A_{\text{new}} = A_1 + A_J \quad (4.1)$$

(4) The shared perimeter P_{sh} between the two regions is found from their adjacency values, according to equation (3.11)

$$P_{\text{sh}} = a_{i,j-1} P_{i,j-1} + a_{j,i-1} P_{j,i-1} \quad (4.2)$$

and the new region perimeter is equal to

$$P_{\text{new}} = P_1 + P_J - P_{\text{sh}} \quad (4.3)$$

(5) The new minimum rectangle coordinates are computed from the old ones, and the new centroid coordinates are given by

$$x_{\text{new}} = (A_1 x_1 + A_J x_J) / A_{\text{new}} \quad (4.4)$$

$$\text{and } y_{\text{new}} = (A_1 y_1 + A_J y_J) / A_{\text{new}} \quad (4.5)$$

(6) The mean and variance of the measured features over the new region are also functions of those of the individual

regions, so that for any feature (intensity or color)

$$\bar{f}_{\text{new}} = (A_1 \bar{f}_1 + A_j \bar{f}_j) / A_{\text{new}}, \quad (4.6)$$

$$\sigma_{\text{new}}^2 = \frac{A_1 (\sigma_1^2 + f_1^2) + A_j (\sigma_j^2 + f_j^2) - A_{\text{new}} \bar{f}_{\text{new}}^2}{A_{\text{new}}} \quad (4.7)$$

Equations (4.6) and (4.7) also apply to the mean and variance of the gradient across the new region

(7) The regions that are adjacent to the new region form the union of the regions that were adjacent to each of the merged regions. The new adjacency values between a region P_K and the new region are computed from the old values as follows

$$a_{\text{new},k} = (a_{1,k} P_1 + a_{j,k} P_j) / P_{\text{new}}, \quad (4.8)$$

$$a_{k,\text{new}} = a_{k,1} + a_{k,j} \quad (4.9)$$

Figure (4.4) is a schematic representation of the merging operation. The rule interpreter identifies the above action as MERGE-REGIONS (*)

(*) Actions recognized by the rule interpreter will be denoted by upper-case characters

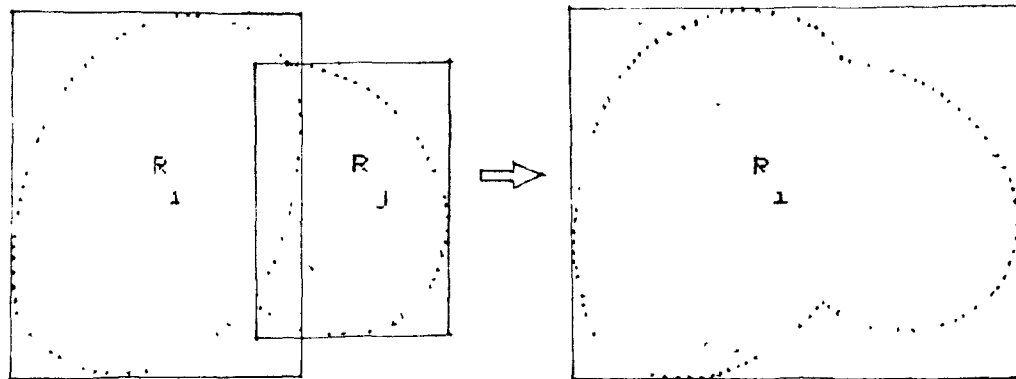


Figure (4.4) : The merging action

4.2.2. Splitting Regions

Unlike merging, the process of splitting a region can be performed in different ways. This is because a region can be divided along any combination of borders, with the sole restriction that the resulting subsets are each composed of connected pixels. Consequently, the outcome will depend on the criteria used to accomplish the splitting. For example, a simple algorithm bisects a region along equally spaced vertical and horizontal lines. A more meaningful method isolates all adjacent pixels that have the same grey level. It is clear that the two processes will generally produce different outputs.

For the rule-based system, two actions provide the general purpose splitting required for this recursive processing methodology. The first is based on the histogram of the measured features across the region, while the second uses a more structural approach that bisects the region along local discontinuities.

4.2.2.1. Histogram Splitting

Histogram analysis has been the basic paradigm of a number of region-based segmentation algorithms [Ohlander, 1975, Nakagawa and Rosenfeld, 1979]. The principal assumption is that isolated peaks within the histogram of one or more features correspond to disjoint sets of pixels in the image. One problem is that histogram information does not reflect spatial relationships among pixels. Therefore, a peak that covers a range of grey level values may, in fact, include contributions from pixels in non-connected image areas.

Because the rule-based system employs merging in addition to splitting, the above problem does not pose a threat to the method of analysis. What is required is a means for disintegrating a region, with the possible prospect of re-merging a subset of the parts during future analysis. A single threshold at one of the feature histograms classifies pixels into subsets with higher and lower feature values. When these are translated back into the image, they create a number (two or more) of connected areas that belong

to one or the other subset. Each disjoint area is given a new region label so that the properties of the region partition introduced in section 3.3.1 are preserved. The situation is depicted by figure (4.5). Errors will result at pixels with values that are close to the selected threshold. A small deviation in feature values will result in the isolation of pixels that should belong to the same region. Because of the similarity in features, the situation is remedied through the future application of merging rules.

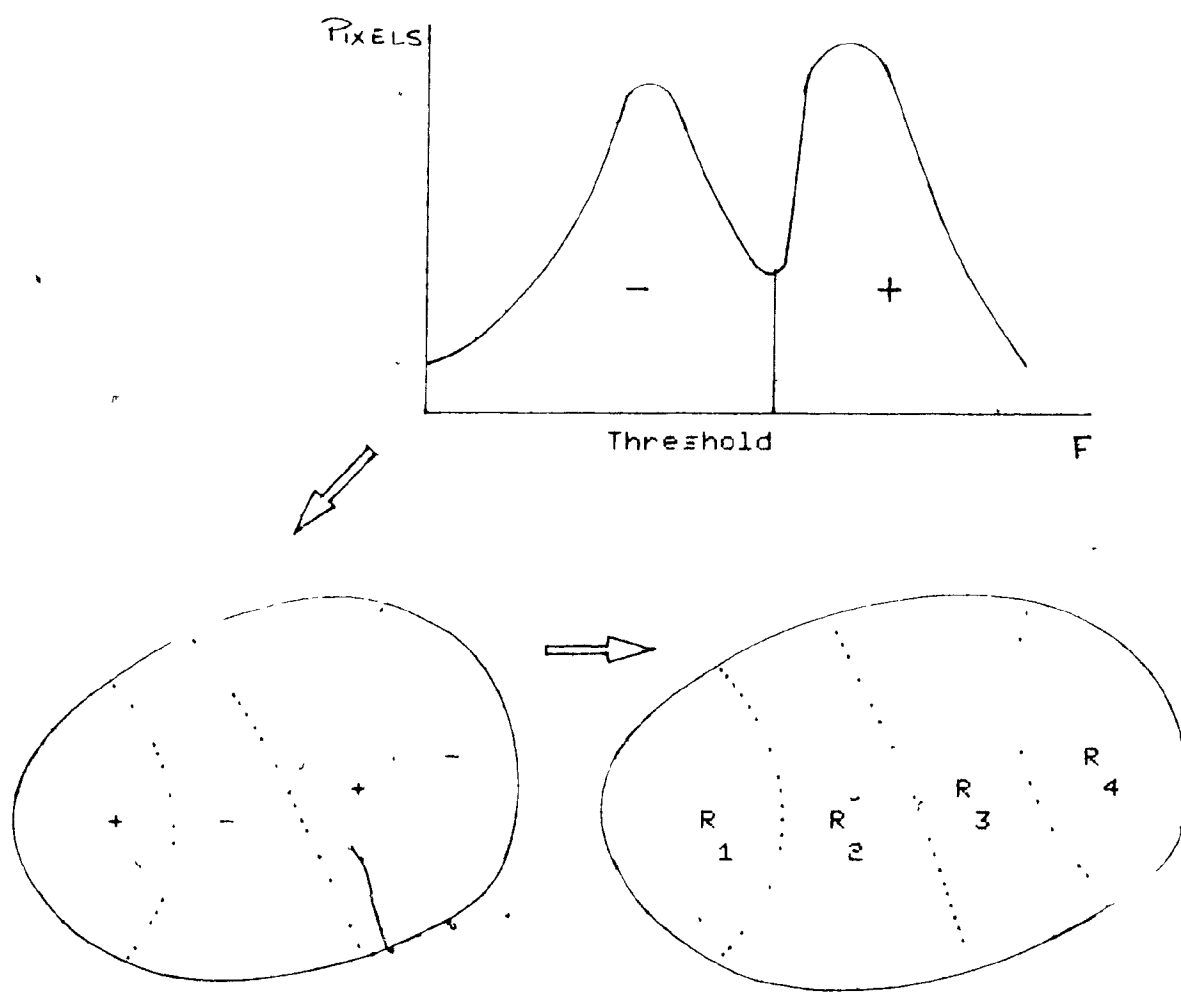


Figure (4.5) Histogram splitting

In order to execute the SPLIT HISTOGRAM action, the REGION ANALYZER must examine the histograms of all measured features, and select the best threshold on one of them. The histograms are computed from contributions of pixels that belong to the original region. Candidates for thresholds are detected at all the valleys between peaks in all histograms. A figure of merit is then computed for each candidate and the one with the maximum value is selected. Evaluation is based on the degree of separability provided by the candidate.

Figure (4.6) shows how a prospective threshold at a typical minimum point T is assessed. Points A and B are the feature values at which the maximum histogram values occur below and above T, respectively. In other words, the two points represent the best peaks on both sides of T. Let the histogram values at the three points be F_A , F_B , and F_T ; then the figure of merit for point T is given by,

$$M_T = \min \left(\frac{F_A - F_T}{F_A}, \frac{F_B - F_T}{F_B} \right) (B - A) \quad (4.18)$$

The first term in equation (4.18) indicates how deep the valley is with respect to the least of the two maxima, while the second term represents the degree of separation between them.

This method for threshold selection is designed to be independent of any tuning parameters, such as those used for peak selection in [Ohlander, 1975]. The approach used here is a relative one. It compares all figure of merit values and selects the feature and the threshold that provide the

best outcome. Another important consequence is that the figure of merit associated with the best threshold represents the degree of separability (or bimodality) of the feature histogram. The latter is used as an argument within rule conditions that determine whether a region is splittable along its histogram, as will be discussed in chapter 5.

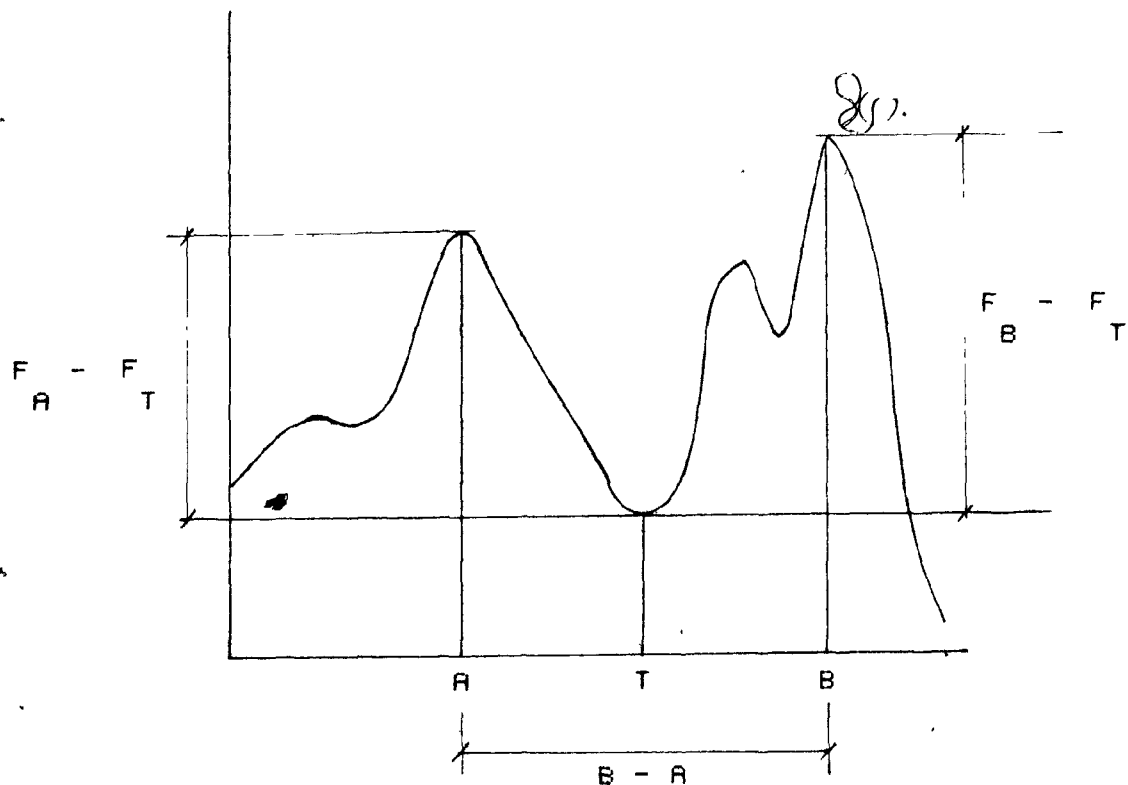


Figure (4.6) Threshold evaluation

4.2.2.2. Splitting at Lines

An alternative method is to split the region at the lines that pass through it. Unlike the previous method where information from all pixels (as reflected in the histogram)

is used, splitting in this case is based on the discontinuity information embedded in the lines. Therefore, this action is associated with rules that attempt to reconcile the region and line partitions.

The simplest situation arises when a single line intersects the boundary of a region at two points, as shown in figure (4.7a). This region can be split into two that are at alternative sides of the line. Splitting is also possible if more than one line intersects the region boundary, as shown in figure (4.7b). If the line does not cross the region boundary an even number of times, as in figure (4.7c), or if the lines are not connected, as in figure (4.7d), then splitting using this method cannot be accomplished. In general, a combination of one or more lines must isolate subsets of the pixels belonging to the region.

The SPLIT AT LINES action is similar to a filling algorithm, in which all adjacent pixels that belong to the original region, and that are not part of a line (i.e., carry the null line label), are given the same region label. As many new labels are used as it takes to cover all the pixels that carried the original region label. This process will split the region into as many disjoint sets as is created by the presence of lines. The only pixels that remain to be classified are those that fall on the lines themselves. These are given the region label of the nearest pixel that belonged to the original region.

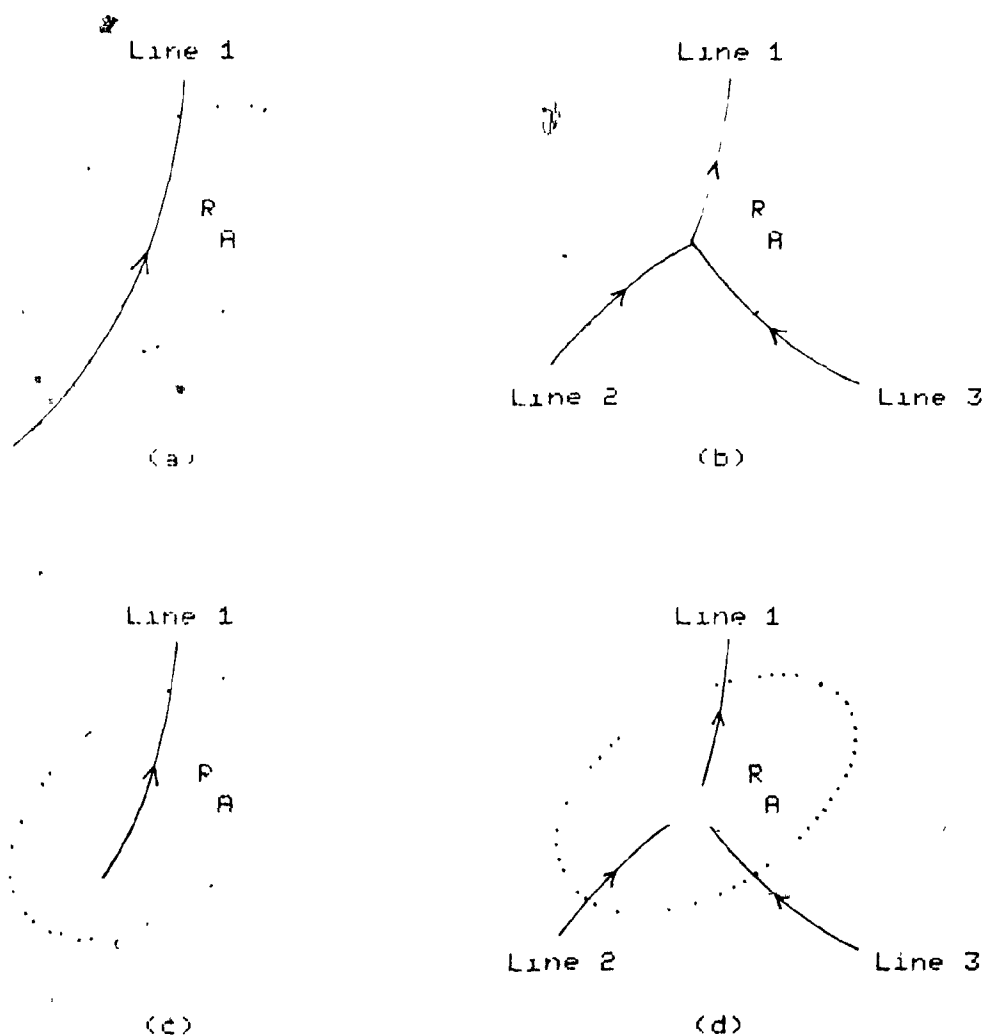


Figure (4.7) Splitting at lines

Both splitting actions described above transform a single region into two or more, and the data structure has to be modified accordingly. However, unlike the merging process described in the previous section, the properties of the new regions cannot be computed from those of the old one. The stored position features are used to locate the new regions, since the latter must fall within the old minimum rectangle coordinates. Otherwise, the statistical, structural, and spatial properties of the new regions are computed by

examining the image, gradient, and region map arrays, as described in section 3.3.1 for newly initialized regions. Because of this, a splitting action is computationally more expensive than its merging counterpart. This conclusion affects strategy setting in terms of assigning priorities to different rules, as will be discussed in chapter 6.

4.3. THE LINE ANALYZER

The line analysis rules are matched against configurations of lines and regions in the STM with the goal of improving the line partition. This is accomplished by removing lines that do not correspond to physical boundaries in the image, by adding lines where gaps exist, and by unifying pairs of lines that belong to the same boundary. The test and modify cycle is executed by the LINE ANALYZER which examines the features that appear in rule conditions and updates them according to the specified actions. The various features have been described in chapter 3, we will now discuss how they are modified by the actions of the LINE ANALYZER.

4.3.1. Deleting Lines

When a line is deleted from the line map, all references to it must be removed from the STM. The following are the steps taken by the LINE ANALYZER to ensure the consistency of the stored data when the DELETE LINE action is executed.

(1) The line is retraced in the line map using its starting point coordinates and its chain code description and all pixels are modified to carry the null label

(2) All references to the deleted line are removed from the spatial relations

(3) The line features are removed from the line description relation

(4) Finally, the chain code description is removed and the line label is freed for future use by newly added lines

It should be noted that updating line information is generally faster than its region counterpart. The main reason is the rapid access provided by the chain code description to all the pixels that belong to one line. On the other hand, a raster scan within the minimum rectangle coordinates must be used to locate all the pixels of a particular region.

4.3.2. Extending Lines

Based on the assumption that an image partition is composed mostly of connected lines, a subset of the rules will attempt to extend open lines. Forward extensions are achieved by adding pixels at the ending point, while backward extensions attempt to elongate the line at its starting

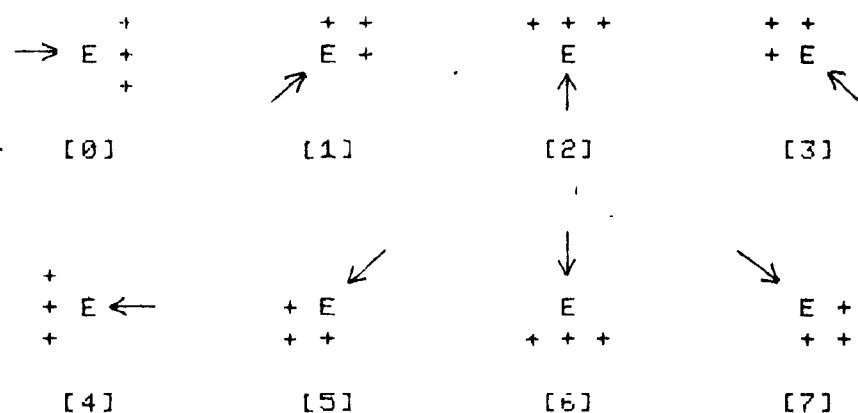
point The LINE ANALYZER uses three different methods for extending lines in both the forward and reverse directions. The methods differ in the criteria that select the new points, as will be discussed in the following sections.

4.3.2.1. Local Gradient Method

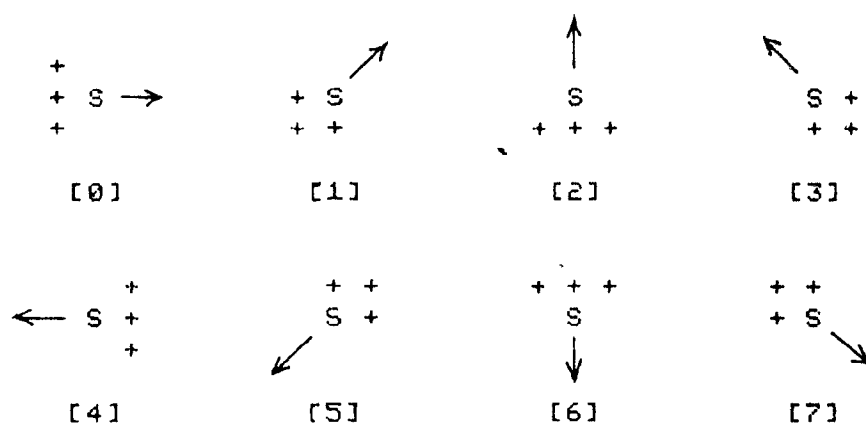
This method extends a line along the direction with maximum local gradient (if any exists) near the terminal point. The gradient magnitude array is examined at pixels adjacent to the terminal point to search for the presence of a local gradient. Only a subset of the eight-connected neighbors are tested. The line direction near its terminal point is used to select possible candidates for addition. In each of the configurations in figure (4.8a), E is the ending point of a line, and the arrow indicates the line ending direction (defined in section 3.3.2.1). Out of the eight neighbors, the three candidates denoted by + signs are selected based on the line ending direction. These are chosen to avoid abrupt changes in direction along the line by allowing a maximum difference of 2 between successive chain code transitions. In a similar manner, figure (4.8b) shows the candidates for backward extension from the starting point S as a function of the starting direction.

Candidates with a LOW gradient value are rejected. If more than one candidate remain, the one with the highest gradient magnitude is chosen as the new terminal point. The process iterates until no new points can be added. This

occurs when another line is encountered, or when all three candidates fail to exhibit a significant local gradient value. The actions that execute the above procedure are termed EXTEND LINE FORWARD and EXTEND LINE BACKWARD.



(a)



(b)

Figure (4.8) Candidates for line extension
 (a) Forward from ending point.
 (b) Backward from starting point.

4.3.2.2. Proximity Method

Gaps are created in boundaries when edge points are missed because they possess weak local gradient. The result is a partition of the physical boundary into several detached line segments. The previous method will not be successful in joining these lines together because of the lack of local gradient evidence. Another action is therefore required that extends a line guided by the presence of another line in the proximity of its open terminal point.

2 The spatial relationships between lines, introduced in section 3.3.2.2, provide the necessary information for such a procedure. When a line has its ending point open, and there exists another line in front of it, the relationship indicates the distance and position of the nearest point on this neighboring line. Again, three candidates are selected based on the ending direction according to figure (4.8a). However, the next point on the line is now selected so as to minimize the distance from the new end point to the nearest point on the line in front, as shown in figure (4.9). The process iterates adding new points to the line until the gap is closed, or the line strays from its destination. The latter will occur if the distance between the two lines increases when a new point is added.

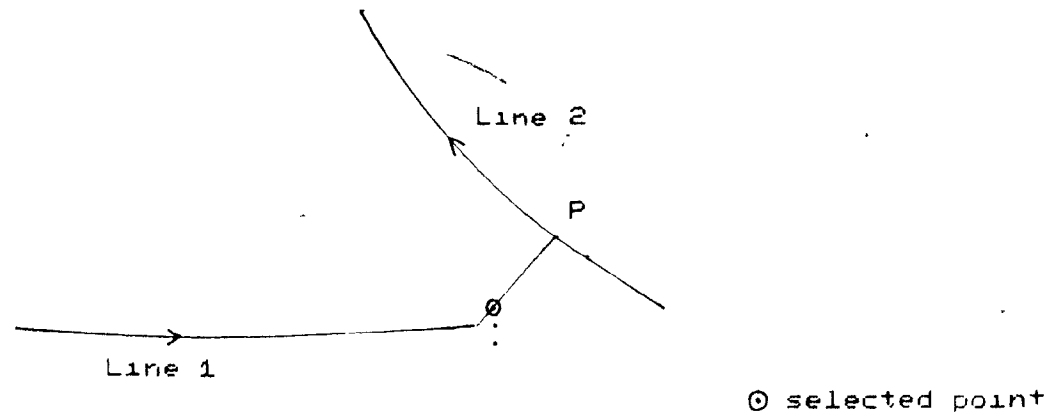


Figure (4 9) Joining two lines

Similarly, a line is joined to another that is behind it by extending its starting position opposite to its starting direction, as long as this reduces the distance between the two lines. The two actions provided by the LINE ANALYZER are termed JOIN LINES FORWARD and JOIN LINES BACKWARD.

4.3.2.3. Extensions Based on Regions

In addition to the previous methods, line extension can be guided by neighboring region information. This is particularly useful when a line falls between two regions and has one or both of its terminals open. In the absence of a significant local gradient and any neighboring lines, the boundary between two different regions provides a path for extending an incomplete line. In this case, more global information is used in improving the line partition.

The basic procedure of selecting the candidates at either terminal follows from figure (4 8). Of these, the pixel that falls at the boundary between two regions is chosen as the new addition to the line, as shown in figure (4 10). The process will continue until the line closes on another line, or an abrupt change in the direction of region boundaries forces all candidates to be non-boundary points. The actions that extend lines by inserting new points between regions are termed INSERT LINE FORWARD and INSERT LINE BACKWARD.

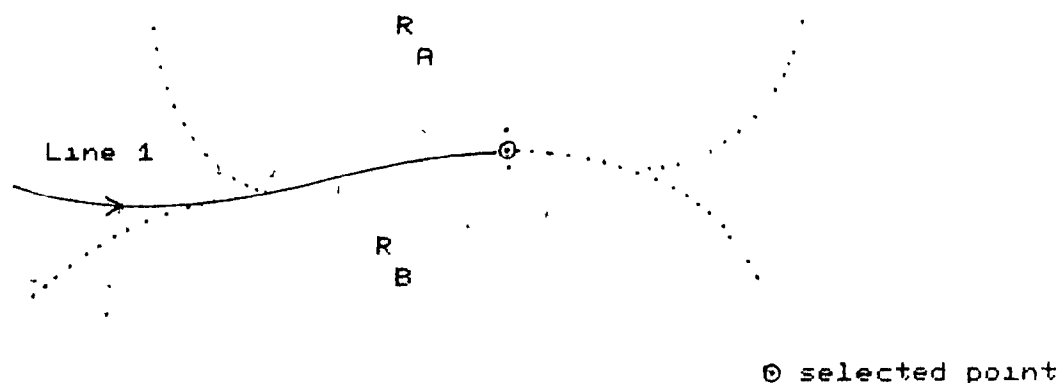


Figure (4 10) Inserting a line at region boundaries

The six actions described in this section will all result in adding new points at the start or end of an existing line. This will require a modification of most of the STM features for that line. The line length is increased and its chain code has new additional entries. The statistical and position features are recomputed, the

starting (or ending) point and direction are changed, and the line label is added to the new points in the map. Finally, the spatial relationships with other lines and regions are altered to reflect changes in distances and length

4.3.3. Merging Lines

When two lines meet, the possibility arises that they are part of the same physical boundary. Figure (4.11) shows two examples in which the above is true. In (a), the two lines are colinear, and can clearly be merged into one line. The situation in (b) is not so straightforward. The lines form a T junction, and if a rule's conditions stipulate it, they can be merged into a combination of two different lines. For example, the illustrated action will minimize changes in the directions of the lines. In general, when two lines are merged, the result is two other lines, one of which can be non-existent (as in figure (4.11a)). In some cases, the second line will be a residual artifact of the merging process, as shown in figure (4.11c). Nevertheless, to preserve the generality of the action, the line is maintained, and will probably be deleted in the future through the application of another rule.

The merging action will dynamically modify the terminal points, lengths, and chain code of the two lines. When a line is merged with another that is in front of it, as in figure (4.12), the starting point of line 1 remains the same, the ending point of line 1 becomes that of line 2, and the

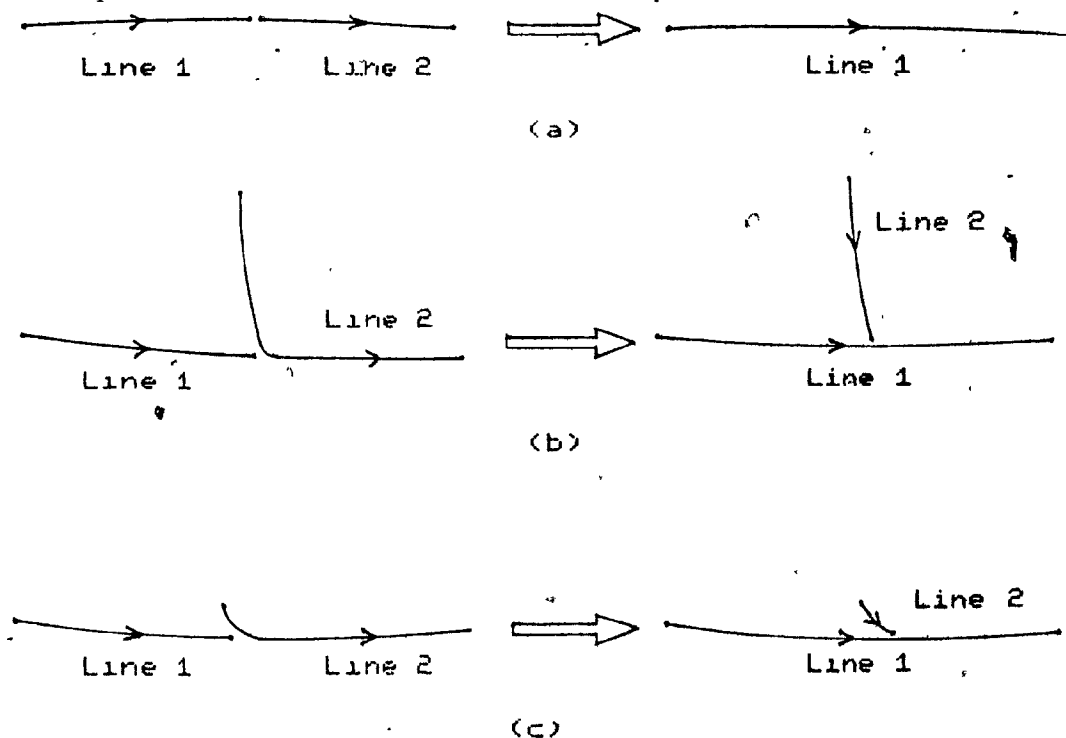


Figure (4.11) Merging Lines

- (a) Two lines merge into one
- (b) A partial merge of two lines into two others.
- (c) The residual line is artifactual, but is maintained to preserve generality.

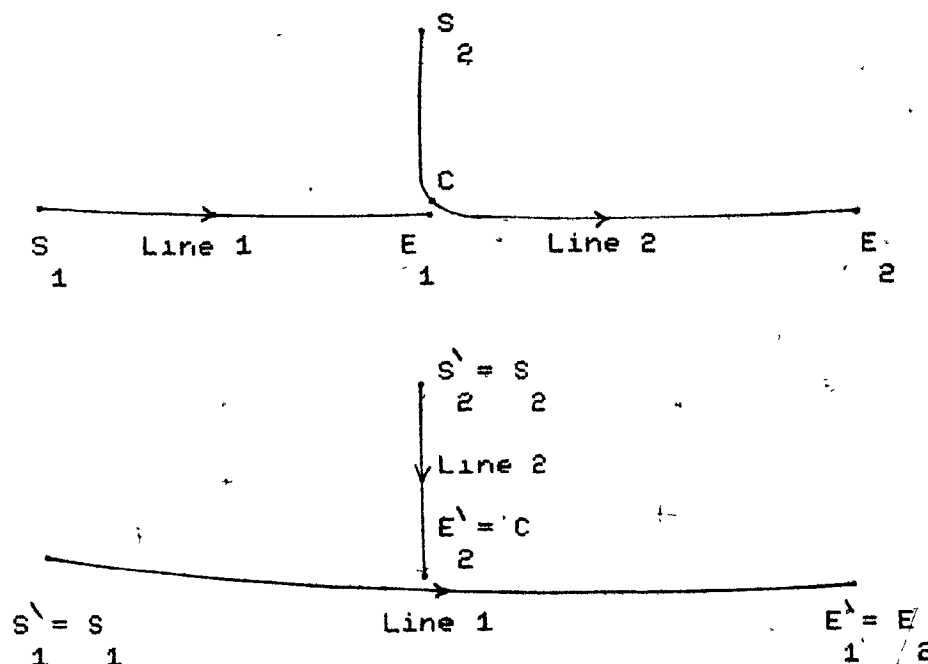


Figure (4.12) The forward merging of two lines.

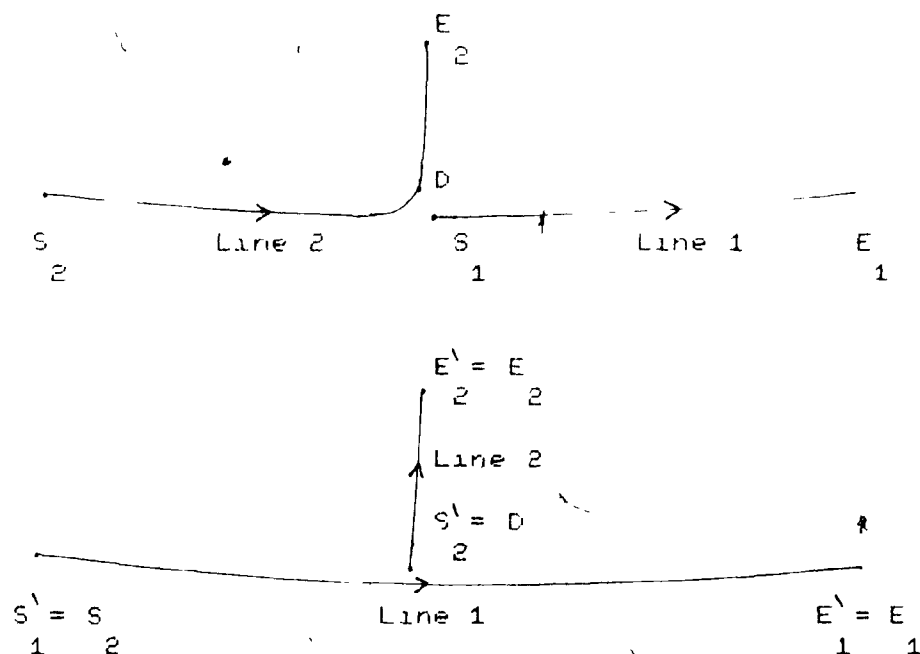


Figure (4 13) The backward merging of two lines

chain code of line 1 is modified to include the part of the chain code of line 2 between point C and its end point. The chain code of line 2 is reduced to the portion between its starting point and point C, which becomes the new end point for that line. The special case in figure (4 11a) will occur when point C is the starting point of line 2, in which case this line will be completely included within the chain code of line 1.

Figure (4 13) depicts the situation when line 1 is merged with line 2 which is behind it. The end point of line 1 remains the same, its starting point becomes that of line 2, and its chain code is shifted so that the portion of chain code between the new starting point and point D is added first. Line 2 now has a new starting point D, the same

ending point and a reduced chain code. Again, line 2 will be completely merged if point D happened to be its ending point.

After the chain codes of both lines have been updated, the remaining STM properties will change considerably. Consequently, the statistical, structural, spatial and position features must be recomputed. Once again we note that given the new chain code representations, this process is done at minimal computational cost. Line rules refer to the above actions as MERGE LINES FORWARD and MERGE LINES BACKWARD, respectively.

4.4. THE AREA ANALYZER

Part of the analysis of an image involves the detection and updating of the focus of attention areas introduced in section 3.3.3. This task is carried out by the AREA ANALYZER module, and includes two types of actions, those that generate new areas and those that maintain existing ones.

4.4.1. Generating New Areas

Areas of attention are generated from groups of regions and lines through the employment of rules that have three types of actions. The first creates a new area when a particular region or line satisfies the criteria for being the nucleus for that area. For example, a large region with low variances in the measured features serves as a nucleus for a smooth area. A small region with a large number of

neighbors becomes the core of a textured area. A large line can form the foundation for a bounded area. When any of the above conditions are met, the AREA ANALYZER creates a new entry in the STM, computes initial descriptive features for the area, and sets up the proper area type. The actions that provide for the above are termed CREATE SMOOTH AREA, CREATE TEXTURE AREA, and CREATE BOUNDED AREA, respectively.

Once the entry for the new area has been established, the second type of action provides for its expansion. Rules examine neighboring regions and lines to determine whether they should be included within the area. If the criteria are met, a region or line is added to the area, and the stored properties are modified to reflect the change. The corresponding actions are termed ADD TO SMOOTH AREA, ADD TO TEXTURE AREA, and ADD TO BOUNDED AREA.

The third and final action type is performed when a growing area meets the criteria for representing a typical entry within its respective category. For example, a smooth area must be of sufficient size, a textured area must contain a minimum number of regions, and a bounded area must contain lines that form a loop. The AREA ANALYZER will then execute an action that identifies the area as an independent entry within the STM and computes its descriptive properties. The action name is either SAVE SMOOTH AREA, SAVE TEXTURED AREA, or SAVE BOUNDED AREA, respectively. The areas are stored in the area description relation described in section 3.3.3, thus allowing them to be used for focus of attention and

other strategy determination functions

4.4.2. Maintaining Current Areas

During processing, the regions and lines that constitute each area undergo changes as a result of merges and splits, as well as additions and deletions. Consequently, the properties of an area, such as the number of regions, the number of lines, and even the shape, size, and average features will vary. These modifications are monitored by the AREA ANALYZER so that if an area ceases to satisfy the criteria set for its type, it can be reclassified accordingly. An existing area can be relabeled into another type, or if it fails to meet the requirements of any type, the area will be deleted. For example, an area is initially classified as textured if it contains a large number of small regions. If these are merged during processing into a smaller number of large regions, the area may become smooth, or it may cease to exist as an interesting entry for focus of attention.

The actions that provide for these adjustments are RELABEL AREA SMOOTH, RELABEL AREA TEXTURE, RELABEL AREA BOUNDED, and DELETE AREA. The first three involve minimal change, since they only require the modification of the area type. The last action will flag the area, free its label, and remove its entry from the list of available areas. This ensures that the stored areas at any point in time, as well as at the end of processing, will truly represent the label

under which they are classified

4.5. THE FOCUS OF ATTENTION MODULE

The four processing modules described above are responsible for the actual processing that produces an output segmentation from an input image. The next two modules deal mainly with the strategy of processing. Because of the complexity of the data structure stored in the STM, a mechanism is required that organizes the methods by which it is tested and modified. The three knowledge modules postulate the presence of a specific region, line, and area to match their rules against at any point during processing. The function of the FOCUS OF ATTENTION MODULE is to choose these data entries from among those stored in the STM.

The selection process proceeds in a hierarchical fashion. From the list of available areas one is designated as the current area of attention. If this list is empty, the whole image is considered as one area. When a new area is selected, the first region and line to be visited are fetched from that area's description relation given in equation (3.23). Other regions and lines are then selected from within the current area by using a number of actions provided by the FOCUS OF ATTENTION MODULE. The latter is responsible for setting up pointers to the selected (current) area, region, and line, as well as their spatial neighbors, as shown in figure (4.14).

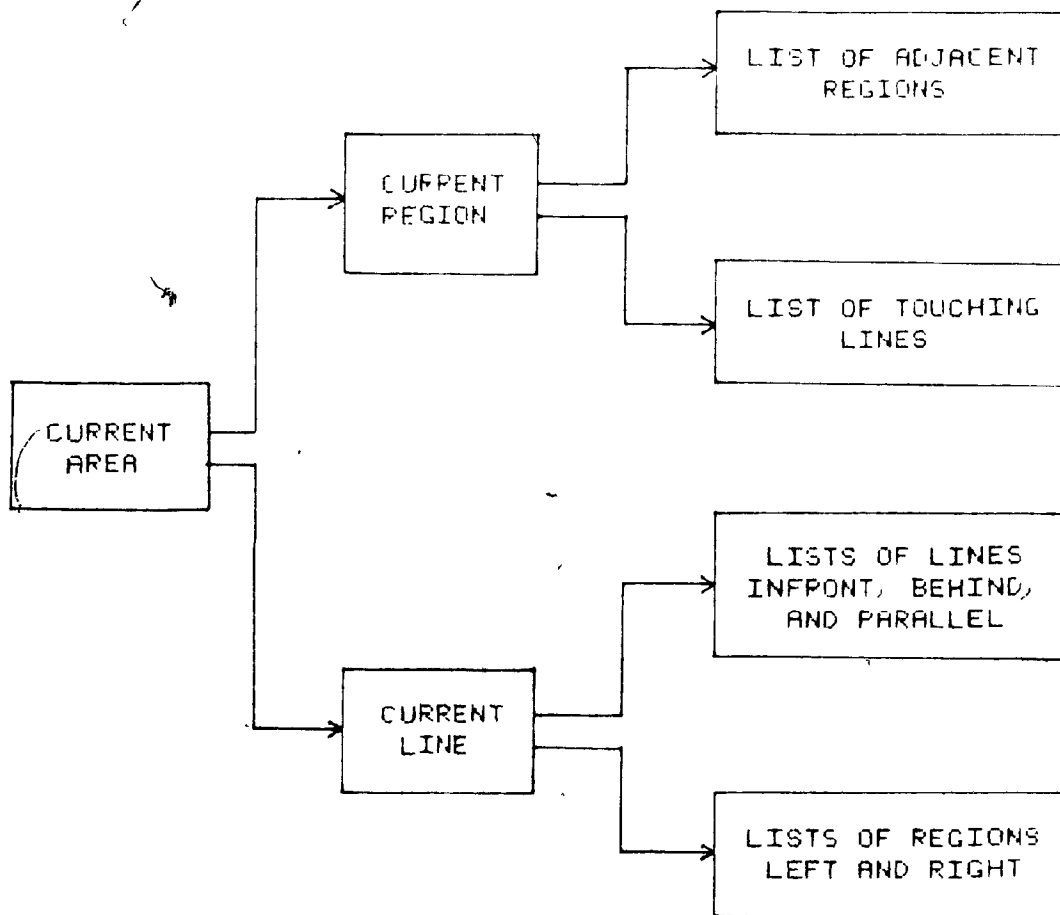


Figure (4 14) Current entries selected by the FOCUS OF ATTENTION MODULE

Whenever the FOCUS OF ATTENTION MODULE is called upon, it will attempt to update the structure shown in figure (4 14). The mechanism is also rule-based and data-driven. Focus of attention rules are matched against the data, and as a result, the current entries may be replaced by new ones. The rules have actions that focus on a new area, as well as others that bring in the next region and next line within the current area. A set of rules can thus be programmed to execute a particular strategy for visiting

the different areas in a given image, and for moving from one region to the next, and from one line to the next, inside each area. The next chapter explains how the focus of attention rules can be employed to execute different strategies, while chapter 6 describes a dynamic mechanism for altering the strategy during processing. Meanwhile, the available options for selecting the next data entry are discussed below.

4.5.1. Focusing on Areas

Different actions are provided that select the next area on which the system will focus its attention. Areas can be chosen based on their type, so that there are actions that GET THE NEXT SMOOTH AREA, GET THE NEXT TEXTURED AREA, or GET THE NEXT BOUNDED AREA. On the other hand, selection may be irrespective of type so that the action will GET THE NEXT AREA from all the available ones.

Each of the above actions will determine the new area, based on factors such as time order or need for processing. The selection criteria and the decision-making process for these actions will be discussed in detail in chapter 6.

Once an area is selected, pointers are set to its features, and to the first region and line in the area, as specified by the description relation of equation (3.23). The FOCUS OF ATTENTION MODULE executes a strategy for visiting all the regions and lines in the area. This

strategy determines the path of processing within the area by specifying a method for selecting the next region and line. Selection methods will be discussed in the next two sections.

To ensure that the path will traverse the whole area the FOCUS OF ATTENTION MODULE must keep track of the regions and lines visited so far. When a new region or line is fetched by one of the actions described in the next two sections, its label is added to one of two lists that compile processing history. Entries on these lists cannot be revisited. This will prevent loops that occur when the mechanism oscillates between two particular entries. Two actions are provided that allow for regions and lines to be revisited by removing their entries from the lists, they are CLEAR REGION LIST and CLEAR LINE LIST, respectively. The actions have the effect of forgetting the previous path of processing within the area.

Finally, the FOCUS OF ATTENTION MODULE has the option of removing from the list of available areas the ones that cease to be of interest. This can be evident if no significant changes take place when an area is brought into processing. To avoid further visits to such an area a rule can mobilize an action that will FREEZE the AREA out of processing. Note that this action is distinguished from the AREA ANALYZER action that deletes an area from the STM. Freezing will render the area inactive in the image, but will maintain its properties for representation in the final outcome. For example, when a textured area ceases to be modified by the

processing modules its entries will not be tested any further, but its features will be noted in the resulting segmentation.

4.5.2. Focusing on Regions

The FOCUS OF ATTENTION MODULE is capable of following alternate paths when visiting the regions within a current area. The available path strategies correspond to actions that move in a different manner from one region to the next. Two possibilities exist: either a random selection of another region in the same area of attention, or a selection based on the features of the current and new regions. The focus of attention rules can execute one of the following:

- (i) Get the region with HIGHEST ADJACENCY value to the current region
- (ii) Get the region with LOWEST ADJACENCY value
- (iii) Get the adjacent region with LARGEST SIZE
- (iv) Get the adjacent region with SMALLEST SIZE
- (v) Get the region that is encountered when scanning the image in a RASTER SCAN starting at the current region
- (vi) Get the region with the next HIGHEST region LABEL.

The first two actions depend on spatial adjacency, the next two on size, the fifth on position, while the last action is independent of the region properties.

Candidates for the next region are arranged according to the required feature. For example, the first action above will sort the spatial neighbors of the current region according to their adjacency values. The regions that are outside the current area are discarded. Of the remaining regions, the one with the highest feature value that has not been visited before (and hence is not on the history list) is chosen to be the new current region. If all the immediate neighbors are on the list, the process will iterate by selecting the neighbor of a neighbor (using the same criterion). The process will only fail if all the regions in the area have been already visited.

Each of the above actions will result in the replacement of the current region. Pointers are set to the properties of the new region and its spatial neighbors, according to figure (4.14).

4.5.3. Focusing on Lines

The determination of which line comes next is the same as in the case of regions, except for the fact that the features used are different. The FOCUS OF ATTENTION MODULE can invoke one of the following actions:

- (1) Get the CLOSEST line that is IN FRONT of the current line
- (11) Get the CLOSEST line that is BEHIND the current line

- (iii) Get the CLOSEST line that is PARALLEL to the current line
- (iv) Get the LONGEST line that is near the current line
- (v) Get the SHORTEST line that is near
- (vi) Get the line with HIGHEST GRADIENT that is near
- (vii) Get the line with LOWEST GRADIENT that is near
- (viii) Get the line in a RASTER SCAN
- (ix) Get the line with the next HIGHEST LABEL

Note that a line that is near another must be in front, behind, or parallel to it. When there is more than one line in front (behind or parallel), the first three options will get the one with the lowest distance from the current line. The next four options, on the other hand, will get the line with a specific feature, with respect to the other lines. Again, each of the actions will update the lists and pointers in figure (4.14).

4.6. THE SUPERVISOR

In addition to determining the sequence in which the data is to be examined and modified, other important control issues remain to be resolved. A mechanism is needed that will specify the order in which the various processing modules are activated, settle the conflicts that occur when more than one rule can match the data, and decide when to end the processing of an image. These functions are undertaken by the SUPERVISOR, a control module that employs a

rule-based, data-driven methodology

The SUPERVISOR matches its own control rules (or meta-rules) on the data. The actions of these meta-rules specify which set of rules should be matched next. Because each of the processing modules is associated with one set of rules, this is equivalent to identifying, based on the data, the next module to be activated. The meta-rule actions are MATCH REGION RULES, MATCH LINE RULES, MATCH AREA RULES, and MATCH FOCUS RULES. In the next chapter we will explain how a set of meta-rules can be used to execute a particular processing strategy.

Other meta-rule actions are executed to commence and terminate processing. One of the functions of the SUPERVISOR is to make sure that a new image is properly initialized. The INITIALIZE REGIONS, INITIALIZE LINES, and GENERATE AREAS actions trigger the appropriate processing modules (the INITIALIZER and the AREA ANALYZER) in order to provide an initial data structure for the rule-based modules. The STOP action is executed only once as part of meta-rules that embody the criteria for terminating the processing of an image.

In addition to the meta-rules, the SUPERVISOR employs a set of higher level rules to adjust the system control strategy by changing the order in which the various rules are matched to the data. A dynamic decision-making process determines the different elements of the control strategy.

The resulting decisions are used by strategy rules which assign priorities to the knowledge rules, the focus of attention rules, and the meta-rules. A complete analysis of the control strategy determination process will be given in chapter 6.

CHAPTER 5

KNOWLEDGE AND CONTROL

5.1. INTRODUCTION

In the past, almost all image segmentation systems have confined their knowledge to an algorithmic type of processing. So far, each of these approaches has provided a solution by incorporating some knowledge about the properties of images, or the scenes that they represent. As a result, a wide variety of low level sources of information about how to segment an image have been accumulating. The structure employed in order to use the knowledge is, in most cases, a simplistic one, that is tailored to the particular needs of the system. Success has been restricted to very special purpose mechanisms that make use of specific knowledge about the class of images under consideration, in order to solve the segmentation problem. General purpose algorithms exist, but they only provide partial solutions, in terms of finding edges, or some other form of incomplete segmentation of an image.

A general purpose solution to the problem must utilize an amalgamation of the learnt experience reported in the literature. For all the knowledge heuristics presented, there will exist a subset that is more suitable for a particular class of images. Moreover, some heuristics will only be useful for parts of an image, while others will be more appropriate in different areas of the same image. A general purpose model must encompass all available information, so that it can bring the proper ones into action whenever the situation arises.

In order to combine all these various sources of information into the same model, we need a format flexible enough to withstand the diversity of its constituents. One important requirement would be the segregation of knowledge from the processing modules, the control mechanisms, and the data to be processed. The addition, modification, or deletion of a piece of knowledge should only require the alteration of the model. This is an imperative necessity, since it renders the model easy to extend and modify.

Knowledge should also be represented in a modular fashion. Each chunk of information should be easily accessible, and as much as possible, independent of the others constituting the model. This requirement arises from the fact that the design of a general purpose knowledge-based system must pass through a learning phase. During this stage, different heuristics are tested in order to evaluate the validity of including them in the model. This should be

modular enough to allow for the employment of an individual heuristic, as well as combinations of these, in such preliminary experiments

The diversity of knowledge sources to be incorporated imposes yet another requirement on the structure of the model. The format used must be general enough to accommodate all the various heuristics presented to it. This must be done without compromising its simplicity. The latter is imposed by the fact that the model will be available to an experimenter, who will use it to interact with the system. The concurrent generality and simplicity constraints suggest the creation of a language that serves as a medium for knowledge representation, as well as an interface between man and machine. Ideally this language should provide the means for translating the linguistic text used by humans into the specific data representation used by a machine.

The requirements described above represent the structural constraints imposed on the model. In addition to the system properties discussed in section 2.4.2, they provide the motivation for using the rule-based approach to image processing. Knowledge about the low level properties of images is encoded into production rules. These are stored in the LTM, and are separated from, but accessible to, the system's processing modules. The LTM thus provides the storage medium for the model, whose modification would only require access to that data structure. The rules are the modular entities of information. They can be accessed,

modified, and replaced individually or in groups. Each rule is written in a symbolic English-like language, and consists of a number of conditions and actions. Conditions are designed to be flexible enough to allow for the representation of all the situations encountered in the data, while actions correspond to procedures provided by the processing modules.

The model stored in the LTM embodies the three levels of rules introduced in figure (1.2). Rules at the first level incorporate information about the properties of regions, lines, and areas in an image of a scene. They describe to the system processes what to expect in the data, as well as what can be done about it. Every rule depicts a particular situation in the image segmentation, and specifies the actions to be executed if that situation arises. These actions will result in the modification of the data in a direction that better conforms to the system's conception of what the output segmentation should be. These rules are therefore termed knowledge rules, since they provide the means for applying the gained experiences to the data. Three sets of knowledge rules will be distinguished, each associated with one of the KNOWLEDGE MODULES. The REGION, LINE, and AREA rules are grouped into these sets according to the data upon which the action of the rule will be performed.

Unlike knowledge rules, the second level of rules is more concerned with how the system will function, rather than what it can do. They provide the control methodology and implement the system strategy in processing an image. Hence, they are termed the control rules. These are, in turn, divided into two classes: the FOCUS OF ATTENTION rules, and the meta-rules. The former determine the next data entry in the STM to be processed. Their actions specify the region, line, or area on which to match the knowledge rules. On the other hand, meta-rules are responsible for specifying the set of rules to activate at any time. In doing so they implement, together with the FOCUS OF ATTENTION rules, a specific control strategy.

The third level contains the strategy rules. They select the processing path strategy to be implemented by the focus of attention rules. In addition, they determine the method by which the meta-rules select the different sets of knowledge rules. The various aspects of the system control strategy will be dealt with in detail in chapter 6. This chapter is concerned with the knowledge and control rules. First, we will describe the structure and symbolic nature of the rules, and introduce the language devised to represent them. This will be followed by a review of the different rules in the model.

5.2. THE STRUCTURE OF THE RULES

Knowledge representation will largely depend on the composition of the rules constituting the model. The previous section summarized the requirements imposed by the data and the system configuration on the structure of the rules. As indicated in section 2.3, the approach used here is that of production systems. A rule has the following format:

CONDITION AND CONDITION.....AND CONDITION ---> ACTIONS (5.1)

The left hand side is composed of a set of CONDITIONS evaluated on the data. The ACTIONS on the right hand side specify particular modifications to the data. The logical ANDs indicate that the action of the rule will only be performed if ALL its conditions are satisfied. Note that there is no provision for a logical OR within a rule. This is because the latter can be accounted for by including another rule that encompasses the alternative situation.

Each rule depicts a certain situation that might be present in the data. A process will match a rule on the features of the regions, lines, or areas, or on some other data item stored in the STM. A search is conducted over the STM for the data configuration portrayed by each rule. If the search is successful, the rule will fire, and its actions will be executed. For this reason, rules have been termed situation-response tuples, or condition-action sets.

5.2.1. The Rule Conditions

The conditions of a rule are logical entities, or equivalently, binary valued variables that can be instantiated to be either TRUE or FALSE. They correspond to logical evaluations or comparisons on the data. Each condition is composed of a logical predicate that may be preceded by the qualifier NOT, a reference to the logical complement of that predicate. The STM contains mostly numerical variables representing features of regions, lines, and areas in the image segmentation. The size of a region and the length of a line are typical examples. Some non-numerical data are also stored. Examples of these include labels that indicate the presence or absence of a certain feature, and those that qualitatively describe a data entry (a line as being a loop, or an area as smooth).

We will distinguish between four classes of logical predicates that can exist in a rule condition. These are classified according to the nature of the logical operation performed, as well as the type of variable that is tested. Following is a discussion of each of the logical predicate classes.

Type 1 Logical Comparisons on Numerical Variables

This logical operation compares two numerical variables together in order to determine whether the first has a value that is LOWER than, EQUAL to, or HIGHER than the value of the second. Examples of this are:

LOWER LINE GRADIENT

HIGHER REGION SIZE

The employment of the qualifier NOT extends the range of operations to include NOT EQUAL, NOT LOWER (equal or higher), and NOT HIGHER (equal or lower)

Type 2 Logical Comparisons on Non-Numerical Variables

Since the variables involved have labels rather than values, a comparison can only yield whether two variables have the same or different labels. This type of condition will thus contain the logical operations SAME and NOT SAME. For example,

SAME REGION LEFT

is a logical comparison on two of the lines in the STM to determine whether they have the same region to their left. Note that this operation is computationally equivalent to the EQUAL operation of type 1

Type 3 Logical Evaluations on Numerical Variables

As described in chapter 3, the STM contains a large number of variables that represent a wide range of features belonging to different data entries. They describe items that are as diverse as size, gray level intensity, spatial relations, and system performance. The task of the rule interpreter is to deal with all of these in a uniform and unified manner. To do this, we define a set of logical operations, that enable the system to access and make use of all stored and computed quantities in one and the same way. Thus, a numerical

A variable is tested in order to determine whether it has a value that is VERY LOW, LOW, MEDIUM, HIGH, or VERY HIGH.

Examples of these include

VERY LOW REGION VARIANCE

MEDIUM REGION ADJACENCY

HIGH LINE LENGTH

Each of the 5 symbols mentioned above defines a range over the scale of each feature. If the value of a feature lies in a given range, the corresponding logical evaluation will be TRUE.

The relative position of the different ranges of symbols is represented schematically in figure (5.1). Note that the ranges for LOW and HIGH include the ranges for VERY LOW and VERY HIGH, respectively. This seems to conform with the linguistic definition of these qualifiers. Other ranges using the scale of figure (5.1) are accessible through combinations of qualifiers and the addition of the NOT qualifier (e.g., NOT LOW, HIGH AND NOT VERY HIGH). The choice of these symbolic qualifiers, their number, and their ranges will be discussed in more detail in the next section.

VERY
|<- LOW ->|

|<----- LOW ----->|

|<----- MEDIUM ----->|

|<----- HIGH ----->|

VERY
|<- HIGH ->|

Figure (5.1) : Relative ranges of symbolic qualifiers.

Type 4 : Logical Evaluations on Non-Numerical Variables

A qualitative variable will assume one of a finite set of labels. An area type, for example, can be SMOOTH, TEXTURE, or BOUNDED. A process name can be REGIONS, LINES, AREAS, FOCUS OF ATTENTION, or SUPERVISOR. Therefore, a logical evaluation on such a variable, would in fact amount to a verification of whether that variable has a particular label assigned to it. In this case, the rule interpreter must be aware of the set of labels associated with each non-numerical variable. Examples of this type of condition include

LINE IS LOOP

AREA IS NOT SMOOTH

PREVIOUS PROCESS WAS REGIONS

IS (ARE) and WAS in the examples above represent the keywords that distinguish this condition type. The use of the NOT qualifier is seen to add more flexibility to the evaluation.

One aspect that relates to the numerical variables in type 2 and 4 conditions is the addition of a DIFFERENCE qualifier. This enables the system to perform all the logical operations cited above on differences in the values of two variables, instead of on the values themselves. Thus "LOW DIFFERENCE IN REGION FEATURE 1" indicates that the difference in the values of the first color feature (average red intensity level) of two regions must be in the LOW range for the condition to be satisfied (the predicate to be TRUE). Each numerical feature must thus have an additional scale as

per figure 5.1. in order to specify the range of the symbolic qualifiers on the differences in that feature. For example, a line length of less than 50 pixels might be considered LOW. On the other hand, the limit for a LOW DIFFERENCE in the length of two lines will necessarily be less than that, maybe 10 pixels. The selection of these ranges will also be discussed in section 5.2.2.

We can now analyse the conditions of our rules and show that they include the following basic items:

(1) A symbolic qualifier that depicts the type of logical operation to be performed on the data. The type of qualifier will depend on the type of feature to be examined, according to the classification given above.

(2) A symbol denoting the data entry on which the condition is to be matched. By default, it refers to the current region, line, or area considered by the system, as specified by the FOCUS OF ATTENTION module. This symbol can be omitted provided that no ambiguities result (*). Other entries that are spatially related to the current ones may also be tested. For example, the features of an adjacent region or a parallel line. Table (5.1) lists all such possible entries, together with the symbols that are used to express them.

(*) The tested operand can only belong to a region, a line, or an area, e.g., line length.

DATA ENTRY	SYMBOL
Current Region	REG
Current Line	LINE
Current Area	AREA
Region adjacent to current region	REGA
Region to the left of current line	REGL
Region to the right of current line	REGR
Line near current line	LINEN
Line in front of current line	LINEF
Line behind current line	LINEB
Line parallel to current line	LINEP
Line intersecting current region	LIN EI

Table (5.1) Possible data entries

(3) A pointer to the feature belonging to the data entry, discussed above (e.g., size, length, and gradient). The value of that feature (a number or a label) is the operand of the logical operation. All available features will be listed in section 5.2.3.

(4) An optional NOT qualifier that negates the effect of the condition.

(5) An optional DIFFERENCE qualifier that forces the operation to be on differences in feature values.

Although it is linguistically difficult to restrict the vocabulary of a language to these basic items, our main concern here is the sufficiency of representing knowledge. A more refined language can be created, and in fact, a larger subset of the English language has been used in a number of artificial intelligence programs [Winston, 1970]. Rather than tailor a language to the needs of a specific application, our approach is to define the constituents that

are required to express the knowledge. A more sophisticated interpreter can be designed to interface any language to the model, by analysing its sentences and filtering out the basic elements cited above. Figure (5.2) summarizes the structure of a condition. Values in square brackets are [optional], while those in round bracket are (variables) to be instantiated.

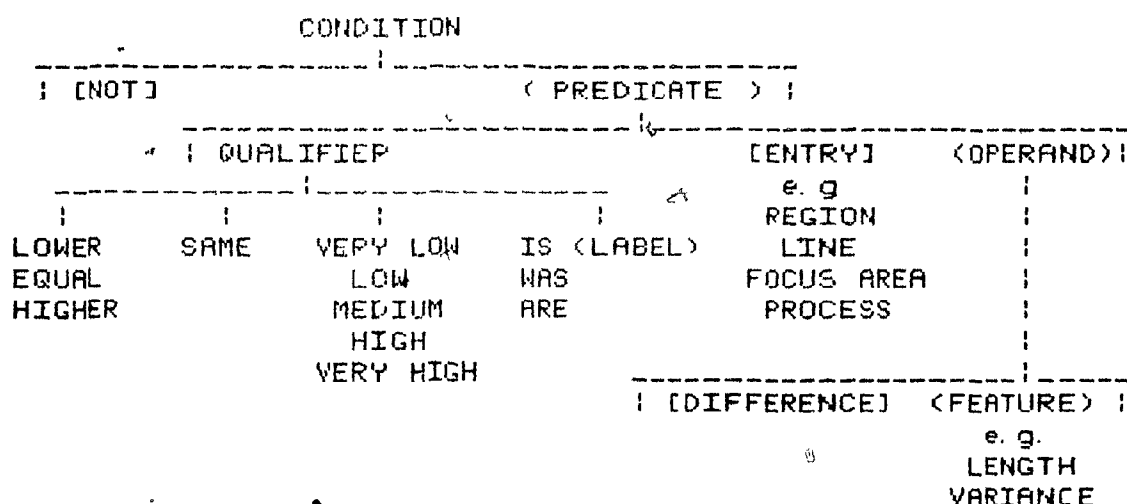


Figure (5.2) The structure of a condition

Note that certain combinations of symbols are illegal. For example, adding the DIFFERENCE qualifier to a non-numerical variable, or testing a non-existent feature of a particular entry (e.g., region length). To deal with sensing this type of situation, we use the RULE INTERPRETER which interfaces the model to the processing modules, as well as to the experimenter. Illegal operations are defined explicitly within the INTERPRETER. This two-way communication channel allows a designer to modify the model.

and provides the means for the model to control the processing modules. As an input channel to the LTM, the INTERPRETER receives commands from a user to edit the model, adding, deleting, changing and examining the rules. It monitors the formulation of the model, and signals any illegal syntactic combinations to ensure an error-free rule set. The output channel then translates the rules for the processing modules. The conditions are analysed in order to specify the type of evaluations to be performed and identify the data items and features to be tested. The processing module then performs the matching operation accordingly. If a match occurs, the INTERPRETER provides pointers to the specific procedures that implements the rule actions. Figure (5.3) is a schematic of the signal flow in and out of the model.

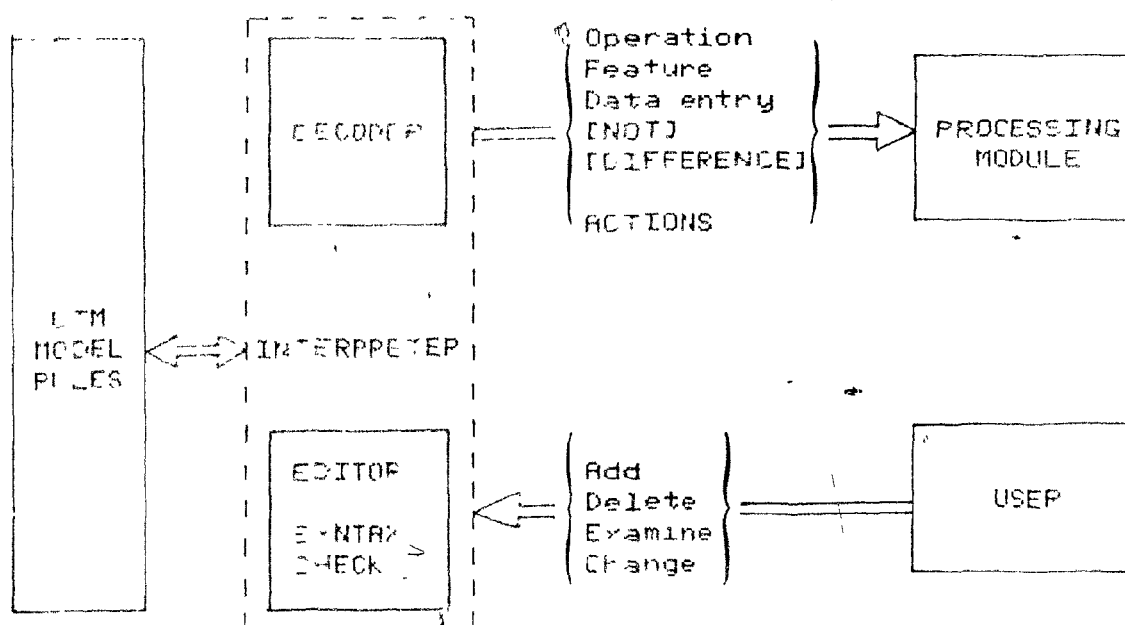


Figure (5.3) The functions of the RULE INTERPRETER

5.2.2. Symbol Manipulation

Part of the rule matching process involves testing numerical values of features to determine whether they satisfy particular criteria. What this amounts to is the verification that a value is bounded by some limits. These define a range over which the required criterion will be valid. For example, it may be necessary to determine whether a region is large. The computational task involves examining the size feature of the region to determine if its value conforms to our definition of "large". Since region size is a numerical quantity expressed in terms of a number of pixels, there must be a range of values over which size is defined to be large. Two values on the scale of that feature will define lower and upper limits for the qualifier. If this is extended to all possible numerical features, as well as all symbolic qualifiers, the number of thresholds defining limits will obviously be unmanageable. The system would suffer from the precise drawback it is trying to avoid by using symbols, the presence of so many parameters to tune. This is a characteristic that has consistently plagued efforts in the field. The inclusion of thresholds for histogram analysis [Ohlander, 1975], region growing [Brice and Fennema, 1970], edge detection [Haralick and Shanmugan, 1973, Robinson, 1977] are but a few examples. These parameters are often problem dependent, and will usually have a direct influence on the output. Their ultimate effect is to reduce the reliability of a system, and render it only useful at a

user interactive level

A more structured approach is needed. Limiting the number of qualifiers that describe each feature is a step in this direction. This, of course, will involve some loss of information, since we are effectively restricting the number of quantization levels. However, this is a welcomed restriction, because it constrains the format for knowledge representation to a level that is compatible with user interaction. There is evidence in every day life that humans use a very small number of levels to quantify the description of events in their environment. A person, for example, is described as being short or tall. The extremes are usually represented by adding "very" to our description. A medium range seems to be reserved for the in-between (or average) values. A choice of five levels will thus provide a basic classification scheme. More fine tuning is sometimes exercised by adding other qualifiers, such as slightly and moderately. Youssef has studied the number of levels to use in order to symbolically characterize the behaviour of a moving blood cell [Youssef, 1982]. His output quantifies the descriptive features into five categories. In order to simplify the process of adjusting the model, we will also limit our representation to the five symbols introduced in figure (5.1).

The second constraint on knowledge representation results from applying the same criteria when dividing the scales of all the features. Once the number of levels has been selected, it remains to define the thresholds that delineate them. A unified way of computing these thresholds will considerably decrease the number of variables. In addition, it transforms threshold selection from being a feature dependent tuning process into a standardized computation that defines symbols.

An extensive study on the selection of the threshold that defines "near" as a function of the maximum measurement was conducted by Denofsky [Denofsky, 1976]. The proposed method was then extended to the case of selecting consecutive thresholds. Basically, the thresholds are defined at multiple values of the scale. If the threshold on "near" is 1, then other thresholds can be found by multiplying it by positive and negative integer powers of 2, as shown in table (5.2).

SYMBOL	RANGE
very near	$x \leq 1/4$
quite near	$x \leq 1/2$
near	$x \leq 1$
not quite near	$1 < x \leq 2$
not quite far	$2 < x \leq 4$
far	$x > 4$
quite far	$x > 8$
very far	$x > 16$

Table (5.2) Consecutive nearness thresholds
(from [Denofsky, 1976]).

The use of this power law is based on the analysis of human performance in descriptive tasks. We will now extend this idea to the selection of thresholds for the symbols related to our features. To do that, we will distinguish between three types of features

(1) Features with an open-ended scale

These are features that have one end of the scale at a well-defined value, while the other end is at a conceptual bound that is not so easily tied to a particular value. A good example of these is the distance feature. The minimum distance between two points is zero. The maximum distance, on the other hand, is not so well defined. A limiting factor is imposed by the fact that we are dealing with a finite image size, but this provides a normalizing factor (e.g., image length + image width) rather than a maximum bound. The scale of such a feature will be termed "open-ended"

The power law can be applied directly to the scale of these features. The thresholds required for our five-symbol scale are placed at intervals that increase by multiples of 2. The resulting divisions are shown in figure (5.4). It is seen that one more constraint is needed in order to find the value of a . This is provided by a restriction on the maximum permissible value for the feature within the context it is used in. Let that value be N , then

$$a + 2a + 4a + 8a + 16a = N, \quad (5.2)$$

$$\text{hence,} \quad a = N / 31 \quad (5.3)$$

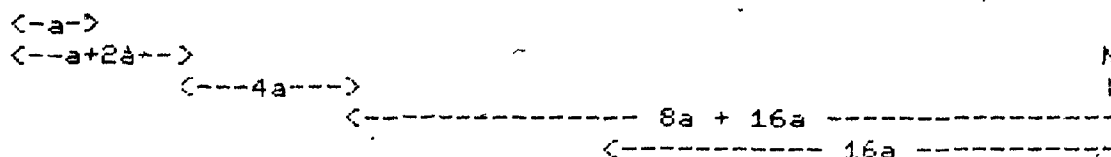


Figure (5.4) Symbolic ranges for features with open-ended scale.

Other features that belong in this class are grey level features (average and variance), region size, line length, and gradient magnitude.

(2) Features that have a closed scale

A closed scale implies that both ends of the scale are well defined. All features that are measured on a relative basis fall into this category. For example, the adjacency between two regions is limited to be between 0 (no adjacency) and 1 (full adjacency or enclosure). Position features are also limited on both sides of the scale, since they are measured relative to the boundaries of the image. These features are characterized by having two reference points to which their scale is tied. They tend to behave symmetrically with respect to each of the end points. A variation of the power law is used to define the scale shown in figure (5.5).

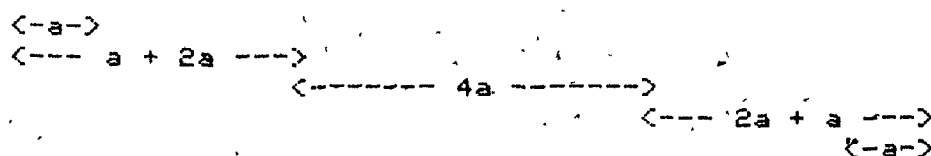


Figure (5.5) : Symbolic ranges for features with closed scale.

Assuming that the minimum and maximum limits are 0 and N , respectively, then

$$a + 2a + 4a + 2a + a = N, \quad (5.4)$$

$$\text{hence,} \quad a = N / 10. \quad (5.5)$$

(3) Features that have a middle reference point.

These features are similar to the ones in the first class in that they have an open ended scale. However, rather than having a maximum attainable value, the constraint on them is imposed by having a fixed reference point that falls in the middle of the scale. This point usually corresponds to a typical value of the feature, or a value that arises from the definition of the feature. An example of this is the aspect ratio, which is a ratio of length to breadth. The reference point is an aspect ratio of 1, when the length is equal to the breadth. Note that aspect ratios of $1/2$ and 2 are qualitatively equivalent, since length and breadth are interchangeable. The power law can thus be applied symmetrically to both sides of the scale, starting at the reference point N , as shown in figure (5.6).

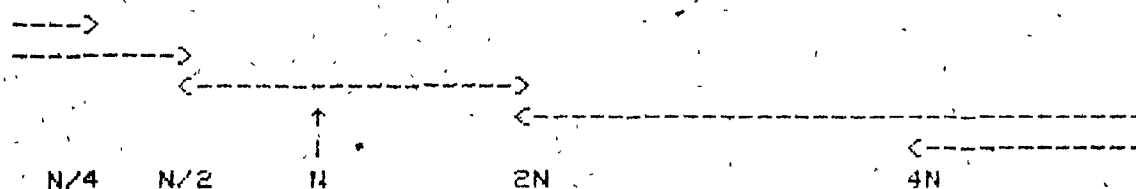


Figure (5.6) Symbolic ranges for features with middle reference point

Note that the scale is left open on both sides. The maximum and minimum attainable values being irrelevant in this case.

Although the above scales have been derived for five symbols, an extension of the above analysis to any number of symbols is quite simple. Youssef has developed a formula that applies to type 2 variables, which automatically selects the thresholds, given the number of symbols used [Youssef, 1982]. Similar formulae are deductible for type 1 and 3 variables.

Four values are thus computed for each feature to define the boundaries between the five symbolic ranges on its scale. Another four values are needed in order to describe the symbolic differences. These can be accounted for by using the same power law. The underlying assumption is that differences are normally an order of magnitude smaller than values. This is made to correspond to a shift of one symbol on the scale. Thus, a LOW DIFFERENCE in size projects into the range of VERY LOW size, MEDIUM DIFFERENCE projects into LOW size, and similarly, HIGH DIFFERENCE into MEDIUM, and VERY HIGH DIFFERENCE into HIGH. The threshold for VERY LOW DIFFERENCE can be computed from the power law as half the threshold for low. Figure (5.7) illustrates the relation between the thresholds for absolute and difference symbol values.

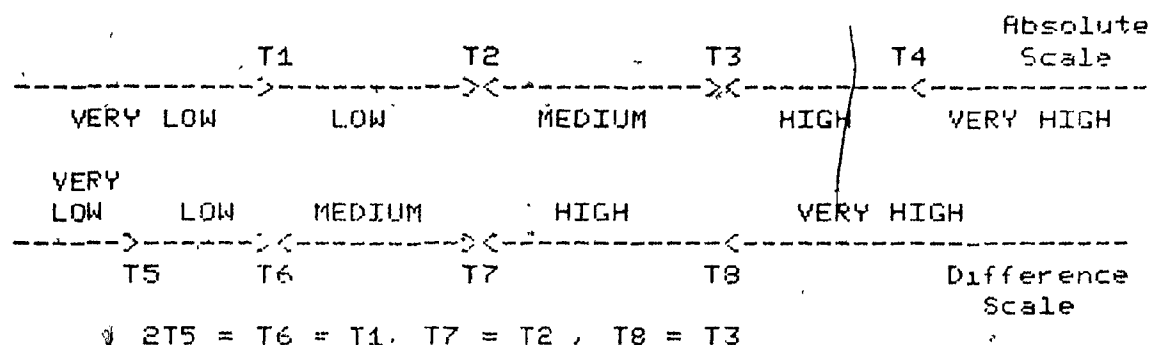


Figure (5.7) . Relation between scales for absolute and difference values

Each feature used in the rules is classified into one of the three classes introduced above. The reference values are identified, and hence, the thresholds that define the absolute and difference symbol scales are computed. Eight such values are determined and stored in a table. This symbol table is a LTM relation that has 9 domains. A tuple is included for each feature having the form:

Feature Name --> T1, T2, T3, T4, T5, T6, T7, T8 , (5.6)

where T1 to T8 are the eight thresholds shown in figure (5.7).

The symbolic model representation introduced in this section is designed to meet two main objectives. First, producing a model that can be easily used by a system experimenter. This is achieved by restricting the number of symbols to a value that is compatible to the limited vocabulary available for linguistic representation. The second goal is to eliminate the dependence on tuning

parameters to which image processing systems seem to be susceptible. By coding the knowledge into a symbolic framework, the problem is reduced to building a model from a well defined and restricted set of constraints.

One primary concern with the symbolic formulation described here is its vulnerability to quantization error, an inevitable consequence of the limited number of symbols. A possible solution to the problem lies in using fuzzy boundaries between the symbols instead of fixed thresholds. Each of the symbols can be represented by a fuzzy set of values. The conditions of a rule would correspond to fuzzy decision-making units that determine how well the criteria are satisfied, as opposed to being binary valued logical predicates. The concepts of fuzzy logic are used in this system for strategy evaluation, and will be described in detail in chapter 6. Their use in the model is only hampered by the complexity they would introduce to the representation. This is the dominant factor during the learning phase of the system, where the model is continuously modified and updated. Once a state of equilibrium is reached, the use of fuzzy logic in formulating the knowledge promises to reduce the sensitivity of the model to quantization errors.

5.2.3. Formulating a Condition

Having described the framework for representing conditions, we can now inspect the ingredients that are available to a designer who wishes to build or modify the model. In order to evaluate the validity of different criteria, the processing modules must examine the data in the STM. A test may directly involve one of the features or data items in the STM, or it may require the computation of a measurement that is a function of the stored data. The question of whether to store a specific quantity, or to compute it when it is needed, is linked to the classical predicament of space versus speed. The norm adopted here is that a measure is precomputed and saved if it is used frequently, and is inexpensive to assess and update. Position features, for example, are continuously employed to locate regions and lines, and are thus evaluated for all data entries at the commencement of processing. On the other hand, the histogram of a region requires large storage space, and is therefore only calculated on request. The distance between the initial and final points of a line is scarcely used, and is again only measured when a condition postulates its assessment for a particular line. The compromise is between performing an a priori computation of a particular feature for all the regions, lines, or areas in the segmentation on the one hand, and repeatedly evaluating it every time it is tested by a rule condition.

Some conditions are thus computationally more expensive than others, an important point to keep in mind when formulating a rule. Simpler conditions should be cited before more complex ones. These are tested in the order in which they appear, and matching will cease as soon as a condition fails. Because a rule will only fire if all its conditions are met, this provides an efficient means of avoiding lengthy and unnecessary computations.

The features that can appear in a condition are reviewed here, and include those stored in the STM (and described in chapter 3), as well as others that are dynamically computed when the condition is tested.

5.2.3.1. Region Features

Table (5.3) summarizes the features that are stored or computed for regions, and are consequently used in conjunction with one of the REGION data entry qualifiers of table (5.1). Descriptive features include the mean and variance of the grey level intensity (or color) features and the gradient over the region. Position is expressed in terms of the coordinates of the centroid and the minimum bounding rectangle. Size, perimeter, circularity, and aspect ratio represent other shape features. The degree of bimodality (if any) of the histogram of the grey level features evaluated over the region (defined in section 4.2.2.1) is used as a splitting criterion.

NUMEPICAL DESCRIPTIVE FEATURES

Feature 1	Feature 2	Feature 3
Variance 1	Variance 2	Variance 3
Intensity	Intensity Variance	Gradient
Gradient Variance	X-Centroid	Y-Centroid
Minimum X	Minimum Y	Maximum X
Maximum Y	Size	Perimeter
Histogram Bimodality	Circularity	Aspect Ratio

NUMEPICAL SPATIAL FEATURES

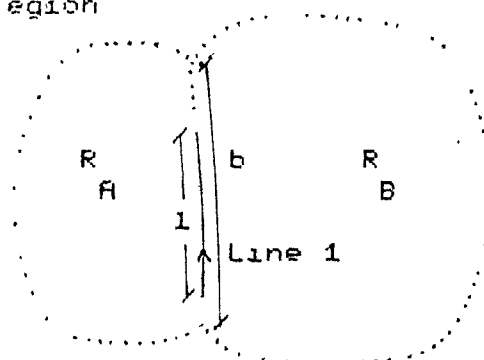
Number of Adjacent Regions	Adjacency Values
Number of Intersecting Lines	Lines between Regions (V)

LOGICAL FEATURES

Histogram is bimodal Region is bisected by line

Table (5.3) Region features

Spatial features relate the current region to adjacent regions and intersecting lines. A value V is computed to indicate the presence of a line between two regions. It is equal to the ratio of the length of the boundary between the regions that is covered by a line, to the total boundary length, as shown in figure (5.8). Two direct logical evaluations are shown. The first applies to the region histogram to determine whether it is bimodal (multi-modal) or unimodal. The second verifies if an intersecting line bisects the region.



$$V = \frac{1}{b}$$

Figure (5.8) Computation of lines between regions.

5.2.3.2. Line Features

The features that are associated with a line are shown in table (5.4). The grey level and position features are analogous to those for regions. A condition must explicitly specify the data entry for which these features are to be tested. Additional position features include the coordinates of the two end points, and the direction of the

NUMERICAL DESCRIPTIVE FEATURES

Feature 1	Feature 2	Feature 3
Variance 1	Variance 2	Variance 3
Intensity	Intensity Variance	Gradient
Gradient Variance	X-Centroid	Y-Centroid
Minimum X	Minimum Y	Maximum X
Maximum Y	Starting X	Starting Y
Ending X	Ending Y	Starting Direction
Ending Direction	Average Direction	Length
Start-End Distance		

NUMERICAL SPATIAL FEATURES

Distance to Line Infront	Nearest Point on Line Infront
Distance to Line Behind	Nearest Point on Line Behind
Distance to Parallel Line	Number of Parallel Points
Adjacency of Left Region	Adjacency of Right Region
Number of Lines Infront	Number of Lines Behind
Number of Parallel Lines	Number of Regions to the Left
Number of Regions to the Right	

LOGICAL FEATURES

Line is open	Line is closed	Line is loop
Line end is open	Line start is open	Line is clockwise

One region to the left
 One region to the right
 Same region to the left and right of line
 Same region left of line 1 and line 2
 Same region right of line 1 and line 2
 Same region to the left of line 1 and right of line 2
 Same region to the right of line 1 and left of line 2
 Two lines are touching (8-connected)

Table (5.4) Line features

line near both ends. The length of a line as well as its average direction are also accessible, and so is the distance between the end points.

The distance between a terminal point on the current line and the closest point on a line that is in front (behind), as well as the position of that close point describe the spatial relation between the two lines (section 3.3.2.2). The number of points a line (partial length) that are parallel to the current line, and the average perpendicular distance characterize the parallel relation (see figure (3.15b)). Other spatial features relate regions to the line being tested.

Logical evaluations are conducted on lines to determine their type. The average orientation along a line is also tested to determine if it travels in a (anti-) clockwise direction. Some spatial configurations are validated by comparing the labels of regions adjacent to two lines (the current line and another that is near it).

5.2.3.3. Area Features

Listed in table (5.5) are the features of focus of attention areas that can be included in a condition. In addition to descriptive features similar to those of regions and lines, performance parameters describe the state of the segmentation within each area. These measures can also be tested in a condition and will be discussed in section 6.3.

NUMERICAL FEATURES

Feature 1	Feature 2	Feature 3
Variance 1	Variance 2	Variance 3
Intensity	Intensity Variance	Gradient
Gradient Variance	X-Centroid	Y-Centroid
Minimum X	Minimum Y	Maximum X
Maximum Y	Size	Total Line Lengths
Uniformity 1	Uniformity 2	Uniformity 3
Region Contrast 1	Region Contrast 2	Region Contrast 3
Line Contrast 1	Line Contrast 2	Line Contrast 3
Line Connectivity	Number of Regions	Number of Lines

LOGICAL FEATURES

Area is smooth	Area is textured	Area is bounded
Area is new		

Table (5 5) Area features

As in the case of lines, logical evaluations assert the area type

5.2.3.4. Other Features in a Condition

Other general features that can be coded in a condition are shown in table (5 6). The number of regions, lines, and areas in the image, as well as their presence or absence, are clues used by the SUPERVISOR in coordinating the system functions. Based on the recent history of processing, the identity of a process in previous cycles can be verified. It is also possible to determine whether the process executed any actions. The usefulness of these tests in implementing control strategies will become apparent later.

The above account reveals a homogeneous approach to the coding of knowledge into rule conditions. The structure of the logical predicates is well defined and its constituents

NUMERICAL FEATURES

Total Number of Regions	Total Number of Lines
Total Number of Areas	

LOGICAL FEATURES

Areas are absent	Regions are absent
Lines are absent	System is starting
Process was Regions	Process was Lines
Process was Areas	Process was Focus
Process was Generate Areas	Process was active

Table (5.6) Other features in a condition

are selected from finite sets of alternatives the qualifier set introduced in figure (5.1), the data entry set given in table (5.1), and the feature set presented in tables (5.3) to (5.6). Features are consistently defined over all the data entries in order to facilitate model formulation. The next section discusses the rule actions.

5.2.4. The Rule Actions

Each rule specifies to a processing module the action to be executed when all its conditions are met. Actions are selected from a finite list that represents the union of the capabilities of the processing modules in modifying the data in the STM. They provide the means for classifying the rules into sets according to the data entry on which each action is performed. A detailed description of how each action is accomplished was given in chapter 4. For completeness, all the available actions are summarized in this section. Table (5.7) provides a complete list of the actions sorted into classes.

AREA ANALYZER ACTIONS

Create Smooth Area	Add to Smooth Area	Save Smooth Area
Create Texture Area	Add to Texture Area	Save Texture Area
Create Bounded Area	Add to Bounded Area	Save Bounded Area
Relabel Area to Smooth	Relabel Area to Texture	
Relabel Area to Bounded	Delete Area	

REGION ANALYZER ACTIONS

Split a Region by Histogram	Merge Two Regions
Split Region at Lines	

LINE ANALYZER ACTIONS

Extend Line Forward	Extend Line Backward
Join Lines Forward	Join Lines Backward
Insert Line Forward	Insert Line Backward
Merge Lines Forward	Merge Lines Backward
Delete Line	

FOCUS OF ATTENTION ACTIONS

Region with Highest Adjacency	Largest Adjacent Region
Region with Lowest Adjacency	Smallest Adjacent Region
Region with Higher Label	Next Scanned Region
Region to the left of Line	Region to the Right of Line
Closest Line Infront	Closest Line Behind
Closest Parallel Line	Shortest Line that is Near
Longest Line that is Near	Strongest Line that is Near
Weakest Line that is Near	Line with Higher Label
Next Scanned Line	Line Intersecting Region
Defocus (Focus on whole Image)	Focus on Areas
Clear Region List	Clear Line List
Freeze Area	Next Area (any)
Next Smooth Area	Next Texture Area
Next Bounded Area	

SUPERVISOR ACTIONS

Initialize Regions	Initialize Lines	Generate Areas
Match Region Rules	Match Line Rules	Match Area Rules
Match Focus Rules	Start	Stop

Table (5 7) : The rule actions

5.2.4.1. Actions on Areas

Some actions create new areas, add regions to existing areas, and compute and store the features of the created areas. Three such actions are defined for each area type. Other actions are performed by the AREA ANALYZER which monitors the focus of attention areas stored in the STM. These include changing the type of an area when its characteristics change during processing, and deleting areas that are not of further use to the focus of attention mechanism.

5.2.4.2. Actions on Regions

The REGION ANALYZER can either merge two regions into one, or split a region into several. Two different methods for splitting a region are given. The first selects a threshold from the histogram of a feature, and finds the connected pixels that are above and below that value. The second bisects a region spatially along the lines that intersect it.

5.2.4.3. Actions on Lines

Actions on lines are more diverse. A line can be extended forward from its last point along the highest local gradient. The same can be done backwards from the line's starting point. Alternatively, a line could be extended forward (or backward) by joining it to another line that is in front of (or behind) it. A third method for extending

lines forward (or backward) is by inserting the label of the line at pixels that are at the boundary of two regions. Two lines can be merged into one line with the same label in two ways. A forward merge will combine the current line with another in front of it, and a backward merge will do the same for the current line and one that is behind it. The two lines must be touching each other. The final action deletes a line completely from the segmentation.

5.2.4.4. Focus of Attention Actions

The focus of attention module possesses different methods for replacing the current region and line by others. The actions listed in table (5.7) are self-explanatory. They should be interpreted as obtaining the next region (or line) that exhibits the particular feature with respect to a current region (or line). This includes, for example, fetching the region that is to the left of the current line, or the line that intersects the current region. Actions that focus on the next area execute a selection strategy that will be discussed in detail in chapter 6. These can be limited to a specific area type, or they can choose any of the available areas.

5.2.4.5. Supervisor Actions

The actions of the meta-rules are different from all of the others in that they do not modify the segmentation data in the STM. They specify which set of rules is to be matched next on the data, and thus indirectly activate the different

processing modules. Other commands include those that trigger the initialization, commence processing, and halt the analysis.

Given the lists discussed above, several conditions can be combined with the appropriate action to produce a particular situation and specify the response. Rules are stored in the LTM in a relation that has $N+M+1$ domains, where N and M are the maximum number of conditions and actions a rule can have, respectively. Each rule is identified by a unique number, and is stored in a single tuple that has the form

Rule No	.	C	,	C	,		,	C	,	A	,	A	,		,	A		(5 7)
		1		2				N		1		2				M		

Rules that have n conditions, where $n < N$, will have the remaining $N-n$ as null (or don't care) conditions. The same applies to rules with less than M actions.

Having described the structure of the rules and reviewed the available conditions and actions, we will now discuss the different types of rules that constitute the model for low level image segmentation.

5.3. THE KNOWLEDGE RULES

The criteria that specify how an image should be segmented are at the heart of our rule-based model. Information about image intensities, region properties, and the existence of lines guide the analysis into creating an

output that is in accordance with the model. By coding these knowledge sources into rules, and applying them to the data, a structured and uniform methodology will result. Other approaches to segmentation use a small number of heuristics in the analysis. This is not due to the lack of applicable heuristics, as is evident by the abundance of efforts in the field, but rather because of the limitations imposed by the extremely rigid format in which the knowledge has been cast. A line detection program, for example, that is built on locating the discontinuities of features in the image array, will hardly produce a perfect (or even acceptable) line drawing. Clearly other heuristics are needed in order to account for deficiencies in the procedure. Methods for improving on the output of the above have been investigated, but they require the execution of other programs that apply new transformations to the image. One may imagine the application of several cascaded procedures to an image, with each one accounting for some knowledge heuristic that was found to enhance the output. This is not an efficient method of analysis. In the course of improving the result, previous accomplishments may be destroyed. Even if this doesn't happen, we still have to contend with a large number of image transformations that yield incremental results which are usually highly localized.

The answer to this problem is to combine all useful processing heuristics into a model, and use it to guide the analysis. This is what a rule-based system accomplishes.

The appropriate piece of information can be brought into the analysis when it is most needed. A specific heuristic will only be used to update that part of the image to which it is applicable. Any number of knowledge sources can be used. The model can be easily tuned by testing, modifying, deleting and adding information. In fact, the main question becomes which heuristics to encode into the rules? Guidelines are needed for selecting the knowledge. Almost half a century ago, psychologists described a set of principles for human visual perception. What has become known as the Gestalt principles amounts to a set of general rules for perceptual grouping [Koffka, 1963]. Although intended for high level cognitive events, these premises also have their consequences in low level context-free applications. One of the factors that has so far limited the use of Gestalt psychology in computer vision is the lack of a framework to represent these logical assertions. The rule-based vision system provides the required mechanism. By modelling the rules after the Gestalt laws, we define guidelines for creating the former and a mechanism for testing the latter. There are five basic principles for grouping visual entities.

(1) Similarity. Similar entities (e.g., in shape or color) tend to form groups, as shown in figure (5.9a).

(2) Proximity. Entities that are spatially close to each other are more likely to be grouped together, as apparent from the example in figure (5.9b).



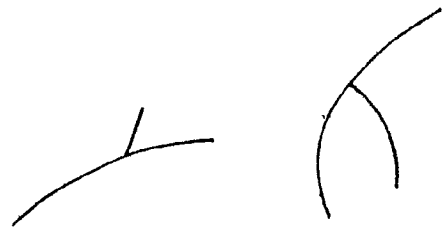
(a) Similarity



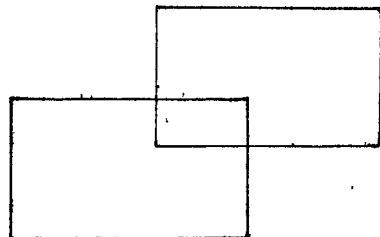
(b) Proximity



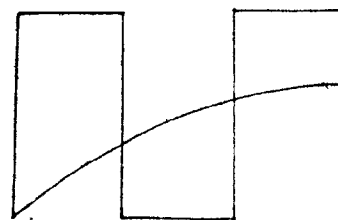
(c) Uniform destiny



(d) Continuity



(e) Closure



(f) Continuity over closure

Figure (5.9) The Gestalt principles of visual grouping

(3) Uniform Destiny (common fate). Elements in these groups appear to undergo homogeneous temporal changes together (e.g. the smooth spatial shift in figure (5.9c))

(4) Good Continuity. Grouping here is based on the optimization of a local feature (e.g., difference in directionality of elements). This principle is also known as the factor of direction or good curve. Examples are shown in figure (5.9d)

(5) Closure. This relates to the inner coherence of the resulting groups, as shown in figure (5.9e)

Other related, although less general factors, include simplicity, symmetry, equilibrium, and good shape. For details on each, the reader is referred to [Wertheimer, 1938, Koffka, 1963, Kohler, 1969]. The outcome of the perceptual process may depend on one or more of the above principles. In certain cases, one factor may be allowed to dominate another, as shown in figure (5.9f) where good curve is favoured over closure.

Our main concern is to encode these logical grouping assertions into the low level model. Each of the segmentation rules may represent one or more of the above principles. In this section, we will present typical examples of each of the three types of knowledge rules, and discuss how the Gestalt criteria are incorporated into the model.

5.3.1. Region Analysis Rules

Rules belonging to this set are concerned with modifying the regions in a segmentation in order to conform with certain criteria. The REGION ANALYZER module applies these rules to the regions stored in the STM. When a match occurs, the process is instructed to either split a region, or merge it with an adjacent one. Thus, the function of this rule set is to specify the merging and splitting criteria to be applied to the present region configuration in order to produce a better partitioning of the image.

The dominant principles in region analysis are those of proximity and similarity. Pixels that are adjacent and have similar intensity (or color) features are grouped into regions. This is the basic principle behind most region growing programs [Levine, 1983]. A general rule merges two adjacent regions (each of which can be a single pixel) as follows [Muerle and Allen, 1968]:

RULE (801)

IF (1) There is a LOW DIFFERENCE in REGION FEATURE 1
 (2) There is a LOW DIFFERENCE in REGION FEATURE 2
 (3) There is a LOW DIFFERENCE in REGION FEATURE 3

THEN (1) MERGE the two REGIONS

This implies that a low difference in the average values of all three color features between the two regions must exist. This general region growing rule will be shown to be responsible for most of the merging that takes place. Alone

however, it cannot account for all the data situations that call upon the region merger. Analysing an image using only this rule as the merging criterion produces many incorrect data configurations. The latter result when non-uniform scene characteristics such as boundaries, shadows, and highlights are subjected to noise, as well as sampling and quantization errors during the image formation process. Based on our understanding of these phenomena, new rules can be added to deal with them on a case-by-case basis. Consider the following rule.

RULE (802)

IF (1) The REGION SIZE is VERY LOW
(2) The ADJACENCY with another REGION is HIGH
(3) The DIFFERENCE in REGION FEATURE 1 is NOT HIGH
(4) The DIFFERENCE in REGION FEATURE 2 is NOT HIGH
(5) The DIFFERENCE in REGION FEATURE 3 is NOT HIGH

THEN (1) MERGE the two REGIONS

The first two conditions describe the situation when a region is of very small size and is mostly surrounded (or totally contained) by an adjacent region. This is an indication that the first region resulted from some noise source. Therefore, a relaxation of the merging criteria in Rule (801) is called for. The last three conditions provide the loosened version that restricts the differences in features to be of low or medium value.

Another aspect to be accounted for is the formation of "hybrid" regions at the boundary between two regions that have high differences in their average intensities (or colors). Due to the graded quantized transition, pixels in the middle acquire a range of intensities that does not allow them to merge with either of their neighbors. Because they are at a boundary, these intermediate regions exhibit a high average gradient, and therefore will be split longitudinally in a direction parallel to the border. This will result in very thin regions along some of the boundaries, that are characterized by low size, a very high (or very low) aspect ratio, and a high average gradient. The merging criteria are again relaxed to eliminate these residual regions, as shown by the following rules:

RULE (803)

IF (1) The REGION SIZE is LOW
 (2) The REGION AVERAGE GRADIENT is HIGH
 (3) The DIFFERENCE in REGION FEATURE 1 is NOT HIGH
 (4) The DIFFERENCE in REGION FEATURE 2 is NOT HIGH
 (5) The DIFFERENCE in REGION FEATURE 3 is NOT HIGH

THEN (1) MERGE the two REGIONS

RULE (804)

IF (1) The REGION SIZE is LOW
 (2) The REGION ASPECT RATIO is VERY HIGH
 (3) The DIFFERENCE in REGION FEATURE 1 is NOT HIGH
 (4) The DIFFERENCE in REGION FEATURE 2 is NOT HIGH
 (5) The DIFFERENCE in REGION FEATURE 3 is NOT HIGH

THEN (1) MERGE the two REGIONS

A similar rule, Rule (805), exists for very low aspect ratio

Because of the limited resolution of the image, faulty regions may also occur if a large number of regions intersect at the same spot. Rule (806) will abolish such errors by allowing a merge with the most similar larger region.

RULE (806)

IF (1) The REGION SIZE is LOW
 (2) The NUMBER of ADJACENT regions is HIGH
 (3) The DIFFERENCE in REGION FEATURE 1 is NOT HIGH
 (4) The DIFFERENCE in REGION FEATURE 2 is NOT HIGH
 (5) The DIFFERENCE in REGION FEATURE 3 is NOT HIGH
 (6) The ADJACENT REGION SIZE is NOT LOW

THEN (1) MERGE the two REGIONS

Two adjacent regions of very small size are also allowed to merge more easily because of diminished contrast sensitivity. The following rule portrays this concept:

RULE (807)

IF (1) The REGION SIZE is VERY LOW
 (2) The ADJACENT REGION SIZE is VERY LOW
 (3) The DIFFERENCE in REGION FEATURE 1 is NOT HIGH
 (4) The DIFFERENCE in REGION FEATURE 2 is NOT HIGH
 (5) The DIFFERENCE in REGION FEATURE 3 is NOT HIGH

THEN (1) MERGE the two REGIONS

The Gestalt principle of similarity is the main factor that motivates the splitting of a region. Then, the resulting regions must consist of pixels that are more similar to each other than the pixels belonging to the original region. Histograms of features are strong indicators of the presence of sub-groups of pixels. A multi-modal histogram signals the need for splitting to

achieve a uniform unimodal distribution of the features in the resulting partition [Tsuji and Tomita, 1973]. In chapter 4, we introduced the method used by the REGION ANALYZER to determine whether a histogram is (at least) bimodal, and if so, compute a measure that is proportional to the degree of bimodality. The corresponding action to split the region was also described. The following pair of rules combine the measurement with the action.

RULE (901)

IF (1) The REGION HISTOGRAM is BIMODAL
THEN (1) SPLIT the REGION according to the HISTOGRAM

RULE (902)

IF (1) The REGION HISTOGRAM BIMODALITY is NOT LOW
THEN (1) SPLIT the REGION according to the HISTOGRAM

The second rule is seen to be more restrictive, since it establishes a lower limit on the degree of bimodality below which splitting will not be allowed. The following rules take into account the size of a region in addition to its histogram.

RULE (903)

IF (1) The REGION SIZE is HIGH
 (2) The REGION HISTOGRAM BIMODALITY is HIGH
THEN (1) SPLIT the REGION according to the HISTOGRAM

RULE (904) -----

IF (1) The REGION SIZE is MEDIUM
 (2) The REGION HISTOGRAM BIMODALITY is NOT LOW
 THEN (1) SPLIT the REGION according to the HISTOGRAM

The two rules provide an example of how the different ranges of symbolic features can be utilized in knowledge representation. Rule (903) demands a highly bimodal histogram for large regions, while this restriction is eased in Rule (904) for medium size regions. Histogram splitting is unreliable (and therefore not allowed) for regions with small (LOW) size because of the increasing effect of the region boundary.

There are other indicators that a region should be split. Rule (905) depicts the case when the variance of the color features is large, indicating a large spread in the feature values from the average.

RULE (905) -----

IF (1) The REGION VARIANCE 1 is HIGH
 (2) The REGION VARIANCE 2 is HIGH
 (3) The REGION VARIANCE 3 is HIGH
 (4) The REGION HISTOGRAM is BIMODAL
 THEN (1) SPLIT the REGION according to the HISTOGRAM

The histogram must still be bimodal, although no restriction is made on the degree of bimodality. This implies that a small dip in the histogram will be sufficient. In fact, the dip is required by the rule action, since a threshold has to be determined for histogram splitting. The next two rules

are seen to use the average gradient over a region as the guide for splitting. A large gradient may be due to a large variance of the features, which is the case already covered by Rule (905). On the other hand, it may be caused by a local discontinuity in the feature values that might not reflect itself in the variances.

RULE (906)

IF (1) The REGION SIZE is HIGH
 (2) The REGION AVERAGE GRADIENT is NOT LOW
 (3) The REGION HISTOGRAM is BIMODAL

THEN (1) SPLIT the REGION according to the HISTOGRAM

RULE (907)

IF (1) The REGION SIZE is NOT LOW
 (2) The REGION AVERAGE GRADIENT is HIGH
 (3) The REGION HISTOGRAM is BIMODAL

THEN (1) SPLIT the REGION according to the HISTOGRAM

Lines may provide evidence for region splitting. Pixels bearing the same region label, but falling on opposite sides of a line passing through, may in fact belong to different regions. The REGION ANALYZER provides an action that splits a region along the lines that intersect it. Rule (908) illustrates how line information is used in processing regions.

RULE (908) -----

IF (1) The REGION SIZE is NOT LOW
 (2) REGION is BISECTED BY LINE
 (3) The LINE LENGTH is NOT LOW
 (4) The LINE AVERAGE GRADIENT is HIGH

THEN (1) SPLIT the REGION at LINES

The use of line information in region analysis is again exemplified in Rule (809). Merging two regions can be inhibited by the presence of a line at their common boundary, even if they have identical average features. This is accomplished by adding another condition to Rule (801) to produce the following

RULE (809) -----

IF (1) There is a LOW DIFFERENCE in REGION FEATURE 1
 (2) There is a LOW DIFFERENCE in REGION FEATURE 2
 (3) There is a LOW DIFFERENCE in REGION FEATURE 3
 (4) The value of LINES BETWEEN REGIONS is LOW

THEN (1) MERGE the two REGIONS

Condition (4) above refers to the fraction of common boundary between the two regions that is covered by lines. This value was defined in figure (5.8). The same condition can be added to other region merging rules.

An evaluation of the region analysis rules presented above is given in chapter 7, when different combinations of rules will be tested and compared. The next section introduces the rules used by the LINE ANALYZER.

5.3.2. Line Analysis Rules

Lines by their nature, provide for a wider variety of data situations. They are also subject to a larger number of actions. In this section, we present examples of various line analysis rules. Unlike regions, similarity is not the main issue here. Instead, other Gestalt principles take on a more prominent role, namely, continuity and good closure. Proximity is of equal importance for lines as it was for regions.

These rules are trying to enhance the initial line configuration that results from detecting the discontinuities in the image intensities. The latter usually produces incomplete lines, as well as faulty lines that are possibly due to noise and therefore must be eliminated. The task is then to discard illegitimate lines and to improve genuine ones. An important clue to both is the presence of a line with one (or both) ends open. Both factors of continuity and closure dictate the necessity to either expand the open end of a line until it encounters another line, or if this is not possible, delete the line.

A credible way to complete lines is by joining an open line to another that is closely IN FRONT of or BEHIND it. The situation is depicted by the following two rules

RULE (1501)

IF (1) The LINE END point is OPEN
 (2) The DISTANCE to the LINE IN FRONT is LOW
 THEN (1) JOIN the LINES by FORWARD expansion

RULE (1601)

IF (1) The LINE START point is OPEN
 (2) The DISTANCE to the LINE BEHIND is LOW
 THEN (1) JOIN the LINES by BACKWARD expansion

The distance constraint is relaxed if the average gradient of the line is not low, or if both lines are colinear (have the same average direction). The corresponding rules are

RULE (1502)

IF (1) The LINE END point is OPEN
 (2) The DISTANCE to the LINE INFRONT is NOT HIGH
 (3) The LINE AVERAGE GRADIENT is NOT LOW
 THEN (1) JOIN the LINES by FORWARD expansion

RULE (1503)

IF (1) The LINE END point is OPEN
 (2) The DISTANCE to the LINE INFRONT is NOT HIGH
 (3) The LINES have EQUAL AVERAGE DIRECTION
 THEN (1) JOIN the LINES by FORWARD expansion

Similar rules exist for joining a line by backward expansion to a line that is behind it.

Region information is also used to aide the joining of two lines. Other constraints can be relaxed if the lines share the same regions to their left and right. This is a strong indication that both lines are part of the same boundary. The following three rules correspond to the data configurations shown in figure (5 10)

RULE (1504) (Figure (5 10a))

```

IF  (1) The LINE END point is OPEN
     (2) The LINE GRADIENT is NOT VERY LOW
     (3) The DISTANCE to the LINE INFRONT is NOT VERY HIGH
     (4) The two LINES have the SAME REGION to the LEFT
     (5) The two LINES have the SAME REGION to the RIGHT

THEN (1) JOIN the LINES by FORWARD expansion

```

RULE (1505) (Figure (5 10b))

```

IF  (1) The LINE END point is OPEN
     (2) The LINE GRADIENT is NOT VERY LOW
     (3) The DISTANCE to the LINE INFRONT is NOT VERY HIGH
     (4) SAME REGION LEFT of LINE 1 and RIGHT of LINE 2
     (5) The two LINES have the SAME REGION to the RIGHT

THEN (1) JOIN the LINES by FORWARD expansion

```

RULE (1506) (Figure (5 10c))

```

IF  (1) The LINE END point is OPEN
     (2) The LINE GRADIENT is NOT VERY LOW
     (3) The DISTANCE to the LINE INFRONT is NOT VERY HIGH
     (4) The two LINES have the SAME REGION to the LEFT
     (5) SAME REGION RIGHT of LINE 1 and LEFT of LINE 2

THEN (1) JOIN the LINES by FORWARD expansion

```

Joining by backward expansion produces three analogous rules for the cases in figures (5 10d) to (5 10f)

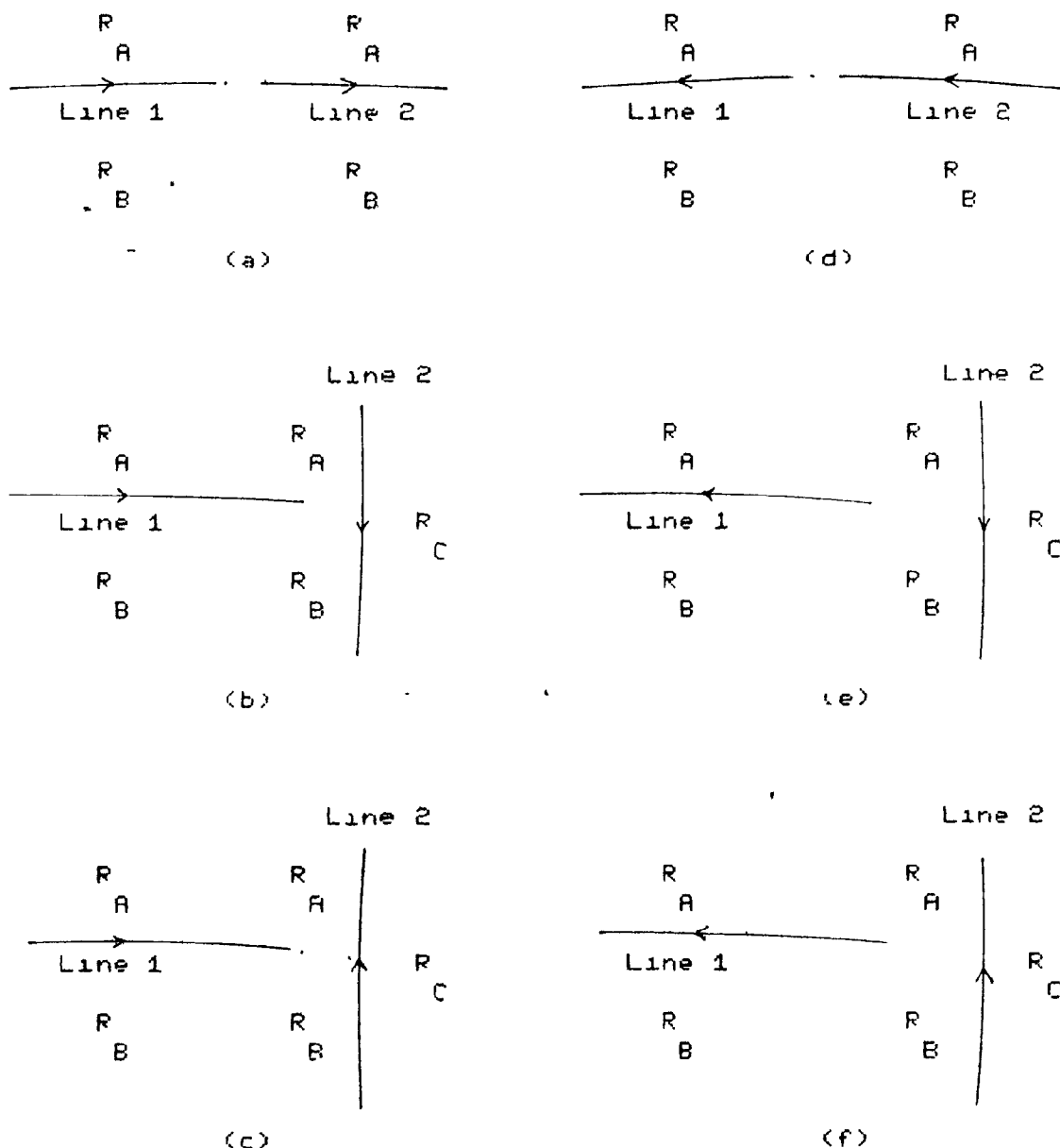


Figure (5 10). Joining lines using region information

In the absence of a close neighbor, an open line can still be extended forward or backward by expanding the appropriate end point along the maximum local gradient at that position in the image. For this to happen, however, a single open line must exhibit strong survival characteristics, both in length and in strength (average

gradient) The following rules represent the above situation

RULE (1507)

```

-----
IF  (1) The LINE END point is OPEN
     (2) The LINE AVERAGE GRADIENT is HIGH
THEN (1) EXTEND the LINE FORWARD

```

RULE (1508)

```

-----
IF  (1) The LINE END point is OPEN
     (2) The LINE AVERAGE GRADIENT is NOT LOW
     (3) The LINE LENGTH is NOT LOW
THEN (1) EXTEND the LINE FORWARD

```

Rules for extending a line backward from its starting point follow easily from the above

The local gradient is also used to expand an open line that has a large number of regions on both sides, as shown in figure (5.11). The line is considered to be at the boundary of two textured areas, and the maximum local gradient is used to define the border. The rule that detects this situation is given by

RULE (1509)

```

-----
IF  (1) The LINE END point is OPEN
     (2) The LINE GRADIENT is NOT LOW
     (3) The NUMBER of REGIONS to the LEFT is NOT LOW
     (4) The NUMBER of REGIONS to the RIGHT is NOT LOW
THEN (1) EXTEND the LINE FORWARD

```

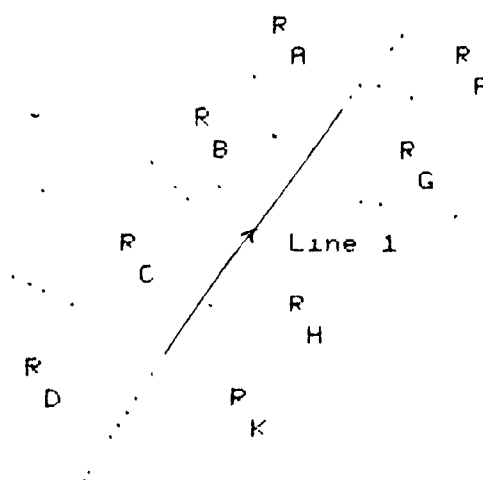


Figure (5 11) Line at the boundary of two textured areas

Rule (1609) will provide for the backward extension in a similar manner

Adjacent region information is also used to expand a line by inserting additional points between two bordering regions. The configuration portrayed in figure (5 12) illustrates how the presence of two different regions, one on each side of an incomplete line, is a strong indication that the line is indeed part of a larger boundary. The following rule is based on this hypothesis

RULE (1510)

```

-----
IF      (1) The LINE END point is OPEN
        (2) The LINE LENGTH is NOT LOW
        (3) There is ONE REGION to the LEFT of the LINE
        (4) There is ONE REGION to the RIGHT of the LINE
        (5) REGIONS to the LEFT and RIGHT are NOT the SAME

THEN    (1) Expand the LINE by INSERTING FORWARD
  
```

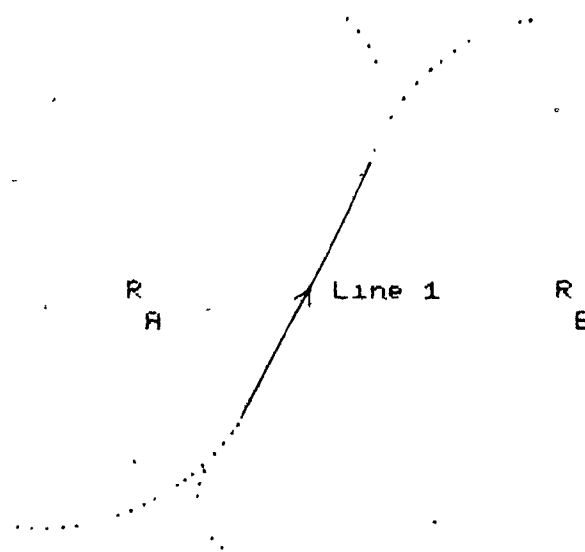


Figure (5 12) Line at the boundary of two large regions

Points are inserted at the starting point of an open line if similar conditions exist

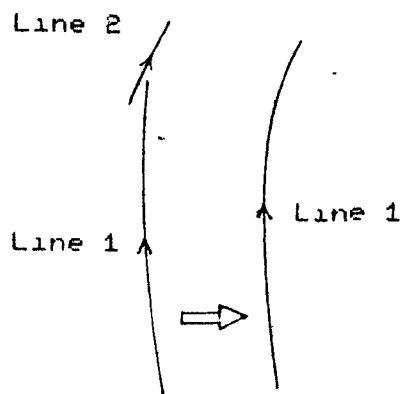
As a result of applying the previous rules, incomplete lines will be extended, and will meet other lines. The resulting junctions may suggest the merging of two lines into one. This happens when both represent different parts of the same boundary, as shown in figure (5 13a). In other cases, merging produces two new lines that form a more appropriate configuration, in terms of the good continuity principle. This situation is depicted in figure (5 13b). Rules are constructed to perform forward and backward line merging, examples of which are

RULE (1701) (Figure (5 13a), case 1)

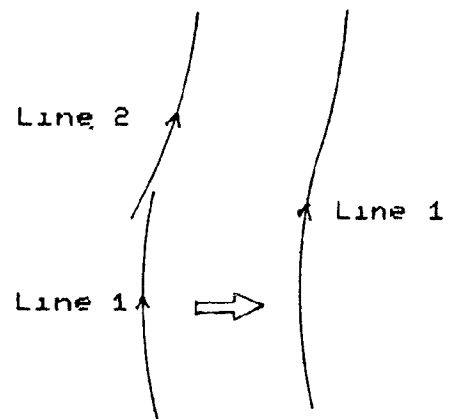
```

-----
IF    (1) The LINE LENGTH is NOT LOW
      (2) The LENGTH of the LINE IN FRONT is LOW
      (3) The LINES are TOUCHING
      (4) The closest POINT IN FRONT is LOW

THEN : (1) MERGE the LINES FORWARD
  
```

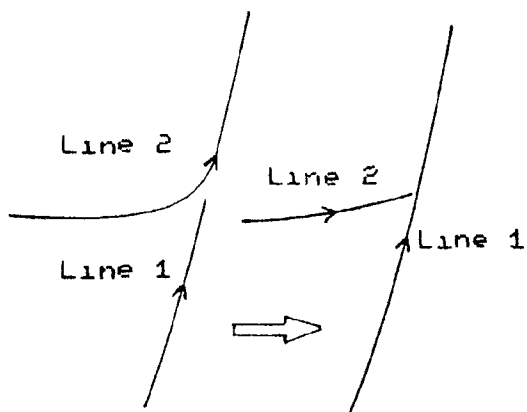


Case 1 Rule (1701)

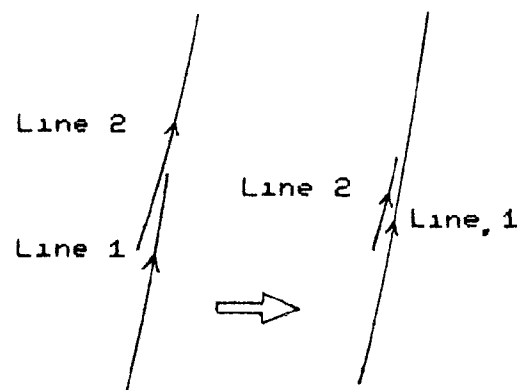


Case 2 Rule (1702)

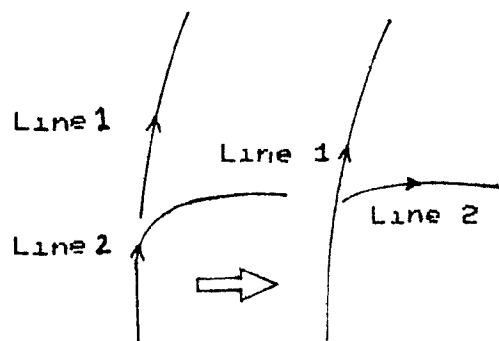
(a)



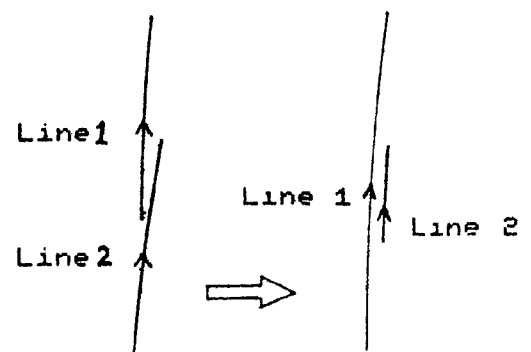
Case 1 Rule (1704)



Case 2 Rule (1705)



Case 3 Rule (1804)



Case 4 Rule (1805)

(b)

Figure (5.13). Rules for merging lines

RULE (1702) (Figure (5.13a), case 2)

IF (1) The LINE LENGTH is NOT LOW
 (2) The LENGTH of the LINE IN FRONT is NOT LOW
 (3) The LINES are TOUCHING
 (4) The closest POINT IN FRONT is VERY LOW

THEN (1) MERGE the LINES FORWARD

The previous two rules represent the situation in figure (5.13a), where the closest point on the second line (the line in front) is near (or at) its starting point. The two rules account for different second line lengths, so that the constraint on the position of the closest point is more stringent if the line length is high (+). A line can also be merged with one that is behind it according to the following rules

RULE (1801)

IF (1) The LINE LENGTH is NOT LOW
 (2) The LENGTH of the LINE BEHIND is LOW
 (3) The LINES are TOUCHING
 (4) The closest POINT BEHIND is HIGH

THEN (1) MERGE the LINES BACKWARD

RULE (1802)

IF (1) The LINE LENGTH is NOT LOW
 (2) The LENGTH of the LINE BEHIND is NOT LOW
 (3) The LINES are TOUCHING
 (4) The closest POINT BEHIND is VERY HIGH

THEN (1) MERGE the LINES BACKWARD

(*) Note that the position of a point on the line is given as a fraction of the line length.

Merging based on similarity of lines is made possible by the following rule

RULE (1703)

IF (1) The LINES are TOUCHING
 (2) The closest POINT IN FRONT is NOT HIGH
 (3) The LINE IN FRONT has a HIGH GRADIENT VARIANCE
 (4) There is a LOW DIFFERENCE in LINE FEATURE 1
 (5) There is a LOW DIFFERENCE in LINE FEATURE 2
 (6) There is a LOW DIFFERENCE in LINE FEATURE 3

THEN (1) MERGE the LINES FORWARD

An analogous rule exists for merging a line with another that is behind it. Note that condition (3) above justifies the splitting of the second line in order to merge part (or all) of it with the first

Merging to improve the continuity of lines, as in figure (5 13b), is achieved through the following rules.

RULE (1704) (Figure(5 13b), case 1)

IF (1) The LINES are TOUCHING
 (2) The closest POINT IN FRONT is MEDIUM
 (3) The LINES END DIRECTIONS are EQUAL
 (4) START and END DIR of LINE IN FRONT ARE NOT EQUAL

THEN (1) MERGE the LINES FORWARD

RULE (1705) (Figure (5 13b), case 2)

IF (1) The LINES are TOUCHING
 (2) The LENGTH of the LINE IN FRONT is NOT LOW
 (3) The LINE START and END DIRECTIONS are EQUAL
 (4) LINE IN FRONT START and END DIRECTIONS are EQUAL
 (5) The closest POINT IN FRONT is LOW
 (6) The DIFFERENCE in AVERAGE DIRECTION is VERY LOW

THEN : (1) MERGE the LINES FORWARD

RULE (1804) (Figure (5 13b), case 3)

IF (1) The LINES are TOUCHING
 (2) The closest POINT BEHIND is MEDIUM
 (3) The LINES START DIRECTIONS are EQUAL
 (4) START and END DIP of LINE BEHIND ARE NOT EQUAL

THEN (1) MERGE the LINES BACKWARD

RULE (1805) (Figure (5 13b), case 4)

IF (1) The LINES are TOUCHING
 (2) The LENGTH of the LINE BEHIND is NOT LOW
 (3) The LINE START and END DIRECTIONS are EQUAL
 (4) The LINE BEHIND START and END DIRECTIONS are EQUAL
 (5) The closest POINT BEHIND is HIGH
 (6) The DIFFERENCE in AVERAGE DIRECTION is VERY LOW

THEN (1) MERGE the LINES BACKWARD

There are a number of cases in which a line should be deleted from the segmentation. The simplest case is when the properties of the line do not justify its existence. Very short line segments that are not connected to any other lines and that exhibit a weak gradient value are considered to be due to noise factors and are consequently removed. This represents an indirect application of the principle of closure, since lines are rarely assumed to end abruptly in an open space. In other words, if a line is not supported by others, and if it does not behave as a plausible independent entity, it will be deleted, as seen from the following rules

RULE (1401)

IF (1) The LINE LENGTH is VERY LOW
 (2) The LINE AVERAGE GRADIENT is LOW

THEN (1) DELETE the LINE

RULE (1402) -----

IF (1) The LINE LENGTH is LOW
 (2) The LINE AVERAGE GRADIENT is VERY LOW

THEN (1) DELETE the LINE

RULE (1403) -----

IF (1) The LINE is OPEN
 (2) The LINE LENGTH is LOW
 (3) The LINE AVERAGE GRADIENT is LOW

THEN (1) DELETE the LINE

A line is also deleted if there is evidence of a multiple representation of a boundary. This commonly occurs in images of natural scenes where the edge between two regions can be several pixels wide. As a result, the process of edge location and boundary tracking will produce a number of parallel lines at the position of the graded boundary. The following rule will eliminate such multiple line occurrences.

RULE (1404) -----

IF (1) The LINE has LOWER LENGTH
 (2) There is a LOW DIFFERENCE in the LINES GRADIENT
 (3) The PARALLEL DISTANCE is VERY LOW

THEN (1) DELETE the LINE

Thus, the shorter of two parallel lines that are very close to each other will be deleted. Note that this rule does not apply to skewed lines (lines that are physically parallel, but have opposite directions). These lines result from the existence of a thin object on a darker or lighter background.

and therefore, should not be eliminated

The configuration of regions in the neighborhood of a line may dictate its removal. This is particularly true when the line falls within a large uniform region, and has a low local gradient as well. A rule that embodies this situation is given by

RULE (1405)

```
IF  (1) The LINE is OPEN
     (2) The LINE GRADIENT is LOW
     (3) The SAME REGION to the LEFT and RIGHT

THEN (1) DELETE the LINE
```

The rules that extend, merge, and delete lines will also be tested and their performance will be evaluated in chapter 7. The third class of knowledge rules, those that analyse areas, is discussed in the next section.

5.3.3. Area Analysis Rules

These include rules that generate areas of attention, and others that update them during processing. The following three rules detect, add to, and save smooth areas in an image segmentation.

RULE (401)

```
IF  (1) The REGION SIZE is HIGH
     (2) The REGION has a LOW VARIANCE 1
     (3) The REGION has a LOW VARIANCE 2
     (4) The REGION has a LOW VARIANCE 3

THEN (1) CREATE SMOOTH AREA
```

RULE (402) -----

IF (1) The AREA IS SMOOTH
 (2) The REGION ADJACENCY is NOT VERY LOW
 (3) The REGION SIZE is NOT LOW
 (4) The REGION has a LOW VARIANCE 1
 (5) The REGION has a LOW VARIANCE 2
 (6) The REGION has a LOW VARIANCE 3

THEN (1) ADD REGION TO SMOOTH AREA

RULE (403) -----

IF (1) The AREA is SMOOTH
 (2) The AREA SIZE is NOT LOW

THEN SAVE SMOOTH AREA

Rule (401) will create the nucleus of a new smooth area from the first large smooth region encountered. Subsequent smooth adjacent regions (if any) are added to the area by Rule (402). Finally, Rule (403) will save the area for further reference during processing, if its size is large enough.

Textured areas are detected in a similar fashion using the following set of rules

RULE (501) -----

IF (1) The REGION SIZE is LOW

THEN (1) CREATE a TEXTURE AREA

RULE (502) -----

IF (1) The AREA IS TEXTURE
 (2) The REGION ADJACENCY is NOT VERY LOW
 (3) The REGION SIZE is LOW

THEN (1) ADD REGION TO TEXTURE AREA

RULE (503) -----

IF (1) The AREA is TEXTURE
 (2) The AREA SIZE is NOT LOW
 (3) The NUMBER of REGIONS is NOT LOW
 (4) The AREA VARIANCE 1 is NOT LOW
 (5) The AREA VARIANCE 2 is NOT LOW
 (6) The AREA VARIANCE 3 is NOT LOW

THEN (1) SAVE TEXTURE AREA

The first two rules set up a search for clusters of small adjacent regions. If the size of a cluster is large enough, and if it exhibits sufficient feature variance, it is saved as a texture area.

Bounded areas are generated by testing the lines in a segmentation. Long lines that close (or almost close) on themselves are sought in order to examine the areas that they encircle. Following are three criteria for detecting such occurrences:

RULE (601) -----

IF (1) The LINE IS LOOP
 (2) The LINE LENGTH is NOT LOW

THEN (1) SAVE BOUNDED AREA

RULE (602) -----

IF (1) The LINE LENGTH is HIGH
 (2) The START-END DISTANCE is LOW

THEN (1) SAVE BOUNDED AREA

RULE (603) -----

IF (1) The LINE LENGTH is MEDIUM
 (2) The START-END DISTANCE is VERY LOW

THEN (1) SAVE BOUNDED AREA

Other area analysis rules monitor existing areas in order to modify their classification according to variations in their properties. Changes are brought about by the merging and splitting of regions, and the addition and deletion of lines. Since the boundaries of an area can be altered in the process, its physical properties may change as well. It is the function of these rules to ensure that areas are consistently categorized according to their features. Examples of rules that change an area's type are

RULE 701 -----

IF (1) The AREA is SMOOTH
 (2) The AREA VARIANCE 1 is NOT LOW
 (3) The AREA VARIANCE 2 is NOT LOW
 (4) The AREA VARIANCE 3 is NOT LOW

THEN (1) RELABEL AREA TEXTURE

RULE (702) -----

IF (1) The AREA IS TEXTURE
 (2) The AREA VARIANCE 1 is LOW
 (3) The AREA VARIANCE 2 is LOW
 (4) The AREA VARIANCE 3 is LOW

THEN (1) RELABEL AREA SMOOTH

Areas are locked out of processing if they are found to be stable enough in terms of their performance characteristics. This allows the system to concentrate on those areas that require further processing. Examples of rules that "freeze" areas are

RULE (703)

IF (1) The AREA is SMOOTH
 (2) The NUMBER of REGIONS is VERY LOW
 (3) LINES are ABSENT
 (4) REGION UNIFORMITY 1 is HIGH
 (5) REGION UNIFORMITY 2 is HIGH
 (6) REGION UNIFORMITY 3 is HIGH
 THEN (1) FREEZE AREA

RULE (704)

IF : (1) The AREA is TEXTURE
 (2) REGION CONTRAST 1 is HIGH
 (3) REGION CONTRAST 2 is HIGH
 (4) REGION CONTRAST 3 is HIGH
 (5) LINE CONNECTIVITY is HIGH
 THEN (1) FREEZE AREA

RULE (705)

IF (1) The AREA is BOUNDED
 (2) LINE IS CLOCKWISE
 (3) ONE REGION RIGHT OF LINE
 (4) REGION UNIFORMITY 1 is HIGH
 (5) REGION UNIFORMITY 2 is HIGH
 (6) REGION UNIFORMITY 3 is HIGH
 THEN (1) FREEZE AREA

An analogous rule to Rule (705) exists for a counter-clockwise line that has one region to its left. Note that an area that is frozen is still available to provide

information about the part of image it represents (e.g. smooth or textured). An area is deleted from the STM only if it no longer carries useful segmentation information. The following rules will eliminate such areas.

RULE (707)

```

IF    (1) The AREA is SMOOTH
      (2) The NUMBER of REGIONS is HIGH
      (3) The AREA SIZE is NOT HIGH
      (4) REGION CONTRAST 1 is HIGH
      (5) REGION CONTRAST 2 is HIGH
      (6) REGION CONTRAST 3 is HIGH

THEN  (1) DELETE AREA

```

RULE (708)

```

IF    (1) The AREA is TEXTURE
      (2) The NUMBER OF REGIONS is LOW
      (3) The AREA SIZE is NOT HIGH
      (4) The REGION UNIFORMITY 1 is HIGH
      (5) The REGION UNIFORMITY 2 is HIGH
      (6) The REGION UNIFORMITY 3 is HIGH

THEN  (1) DELETE AREA

```

RULE (709)

```

IF    (1) The AREA is BOUNDED
      (2) The NUMBER OF LINES is HIGH
      (3) The LINE CONTRAST 1 is HIGH
      (4) The LINE CONTRAST 2 is HIGH
      (5) The LINE CONTRAST 3 is HIGH

THEN  (1) DELETE AREA

```

This concludes our discussion of the knowledge rules for low level image segmentation. From the analysis, we can conclude that the Gestalt principle of similarity plays an important role in region processing. On the other hand,

closure and continuity are more prominent in line processing. Proximity is an essential factor in grouping both regions and lines. However, the principle of uniform destiny does not find an application at the low level of processing, since it seems to involve high level assertions about the grouped perceptions.

The knowledge rules presented in this section are typical examples of those used by the system. A complete listing of the knowledge rules is given in Appendix A.

5.4. THE CONTROL RULES

The focus of attention rules and the meta-rules differ from the knowledge rules in that they do not directly modify the segmentation data. Yet, their presence is essential to the functioning of the system. They control data selection and rule matching and are hence called control rules. There are numerous ways in which to execute any of these tasks. Therefore, a specific control strategy must define the exact method for choosing data items and knowledge rules. In chapter 6, we will introduce a dynamic data-driven method for selecting the appropriate control strategy. Meanwhile, in this section, we describe how the focus of attention rules and the meta-rules are used to implement various strategies.

5.4.1. The Focus Of Attention Rules

The FOCUS OF ATTENTION process performs two levels of data selection. An area is selected, and then the regions and lines within the area are chosen. When all have been visited, the process will move to a new area. The whole procedure is repeated until all the areas in the image are covered.

The transition from one area to the next is accomplished when the focus of attention process fails to select a new region or line in the current area. This failure is recorded in the SYSTEM STATUS relation of equation (3.33). In this particular cycle, the process is marked as INACTIVE. When this happens, the FOCUS OF ATTENTION process must be invoked again to get a new area. If a failure occurs now, it will signal that all the areas have been visited. An option is provided to either stop the processing, or revisit the areas.

By examining the history of processing in the previous two cycles (stored in the STATUS relation), different conclusions can be arrived at. Table (5.8) lists the different cases and their interpretations (N/A correspond to don't care conditions).

In case A, a new region or line has been fetched. The FOCUS OF ATTENTION process was marked as ACTIVE, and now a KNOWLEDGE MODULE can be invoked to test its rules. The selection of new regions and lines is accomplished by matching the general rule.

Case	Cycle	Process	State	Interpretation
A	Next	Previous	N/A	A new region or line is fetched
	Previous	FOCUS	ACTIVE	
B	Next	Previous	NOT FOCUS	All regions and lines in current area were visited
	Previous	FOCUS	INACTIVE	
C	Next	Previous	FOCUS	A new area was fetched but no regions or lines are selected
	Previous	FOCUS	INACTIVE	
D	Next	Previous	FOCUS	All areas have been visited
	Previous	FOCUS	INACTIVE	

Table (5.8) Interpretation of processing history

RULE ("1XX")

IF (1) The PREVIOUS PROCESS was NOT FOCUS

THEN (1) " GET THE NEXT REGION "

(2) " GET THE NEXT LINE "

The quotations in the rule number and in the actions indicate that they are variables to be instantiated. In sections 4.5.2 and 4.5.3, actions that correspond to different methods of obtaining the next region and the next line have been introduced. A specific strategy must select the method that is most suitable for the properties of the current area. This is another point to be discussed in detail in chapter 6. Meanwhile, it will suffice to say that different rules are

included in the LTM, each with particular region and line selection methods (e.g. get the largest adjacent region and the closest line). However, only one of these rules will be made available to the FOCUS OF ATTENTION module at any point in processing. When this rule is matched, it will fetch the next data entries according to the specified strategy.

Cases B and C in table (5.8) correspond to the situation where the current area cannot contribute any new regions or lines, and therefore a new area is now called for. The following two rules are seen to be in accordance with the two cases.

RULE (101) (Case B)

IF (1) The PREVIOUS PROCESS was FOCUS
 (2) The PREVIOUS PROCESS was NOT ACTIVE
 (3) The NEXT PREVIOUS PROCESS was NOT FOCUS

THEN (1) GET THE NEXT AREA

RULE (102) (Case C)

IF (1) The PREVIOUS PROCESS was FOCUS
 (2) The PREVIOUS PROCESS was NOT ACTIVE
 (3) The NEXT PREVIOUS PROCESS was FOCUS
 (4) The NEXT PREVIOUS PROCESS was ACTIVE

THEN (1) GET THE NEXT AREA

The selection of the next area will be shown in chapter 6 to be based on a data-driven decision-making methodology.

Case D in table (5 8) occurs when all the areas in the image have been visited. The next rule allows for a second path over the areas by erasing the history lists for regions and lines (see section 4 5 1)

RULE (103) (Case D)

IF (1) The PREVIOUS PROCESS was FOCUS
 (2) The PREVIOUS PROCESS was NOT ACTIVE
 (3) The NEXT PREVIOUS PROCESS was FOCUS
 (4) The PREVIOUS PROCESS was NOT ACTIVE

THEN (1) CLEAR REGION LIST
 (2) CLEAR LINE LIST

In the next section, we will describe a meta-rule that uses the same conditions to stop the processing

The previous rules execute a specific strategy for visiting the areas, regions, and lines in the image. In addition, other rules can override these and fetch a region or line other than those specified by the default strategy. This is particularly useful when an interesting data configuration is detected that requires immediate investigation. The following is an example of such a rule.

RULE (201)

IF (1) The LINE GRADIENT is HIGH
 (2) The LINE LENGTH is HIGH
 (3) SAME REGION LEFT and RIGHT of the LINE

THEN (1) GET the REGION to the LEFT of the LINE

In this rule, a line that has a high gradient and length intersects a region. Thus, the latter should be examined to

determine whether it should be split. Another interesting case occurs when a line encircles a region, as given by the following two rules

RULE (202)

IF (1) The LINE is LOOP
 (2) The LINE is CLOCKWISE
 (3) The RIGHT REGION ADJACENCY is HIGH

THEN (1) GET the REGION to the RIGHT of the LINE

RULE (203)

IF (1) The LINE is LOOP
 (2) The LINE is NOT CLOCKWISE
 (3) The LEFT REGION ADJACENCY is HIGH

THEN (1) GET the REGION to the LEFT of the LINE

If the average gradient of a large region is high, this may be due to a line passing through it. The following rule brings any such line to the attention of the system.

RULE (204)

IF (1) The REGION SIZE is HIGH
 (2) The REGION AVERAGE GRADIENT is HIGH

THEN (1) GET the LINE that INTERSECTS the REGION

Parallel lines also provide a configuration worth investigating, as is evident from the next rule:

RULE (205)

IF (1) The PARALLEL DISTANCE is NOT VERY LOW
 THEN (1) GET the PARALLEL LINE

Parallel lines that are very close are subject to deletion according to Rule (1404) (see section 5.3.2).

Rules (201) to (205) are given higher priority than the default rule ("1XX") in fetching the next region or line. If none of them matches the default strategy is executed as described at the beginning of this section.

5.4.2. The Meta-Rules

The SUPERVISOR coordinates the activities of all the other processing modules within the framework of a specific control strategy. A strategy defines the priorities among the different rule sets. By matching the meta-rules to the data, and in particular, the system-related data stored in the STATUS relation of equation 3.33, the SUPERVISOR determines the set of rules to be tested (and hence the process to test them). The simplest examples of meta-rules are those that ensure proper initialization, as follows:

META-RULE (1)

IF (1) REGIONS ARE ABSENT
 THEN (1) INITIALIZE REGIONS

META-RULE (2)

IF (1) LINES ARE ABSENT
THEN (1) INITIALIZE LINES

META-RULE (3)

IF (1) AREAS ARE ABSENT
THEN (1) GENERATE AREAS

META-RULE (4)

IF (1) The previous PROCESS was GENERATE AREAS
(2) The previous PROCESS was ACTIVE
THEN (1) GENERATE AREAS

META-RULE (5)

IF (1) The previous PROCESS was GENERATE AREAS
(2) The previous PROCESS was NOT ACTIVE
THEN (1) Match the FOCUS OF ATTENTION rules

The first two rules will initialize the region and line maps according to the procedures given in section 4.1. The next two rules will recursively generate areas of attention based on the initial region and line configurations, and the rules discussed in section 5.3.3. When all the areas have been created, the FOCUS OF ATTENTION is called upon to match its rules and obtain the first data entry to process.

During processing, other meta-rules invoke the FOCUS OF ATTENTION process whenever a new data entry is sought. Because of the two levels of data selection discussed in the previous section, these rules must allow the focusing process to fail twice before concluding that all the data entries have been exhausted. This corresponds to case Q in table (5.8). Accordingly, Meta-rule (6) provides a default stopping criterion for the system.

META-RULE (6)

IF (1) Previous PROCESS was FOCUS
 (2) Previous PROCESS was NOT ACTIVE
 (3) Next previous PROCESS was FOCUS
 (4) Next previous PROCESS was NOT ACTIVE

THEN (1) STOP

Otherwise, a single failure of the FOCUS OF ATTENTION process to obtain a new entry signals the exhaustion of the regions and lines within the current area. The following rules are used to re-invoke the focus of attention rules in order to obtain a new area.

META-RULE (7)

IF (1) Previous PROCESS was FOCUS
 (2) Previous PROCESS was NOT ACTIVE
 (3) Next previous PROCESS was NOT FOCUS

THEN (1) Match the FOCUS OF ATTENTION rules

META-RULE (8)

IF (1) Previous PROCESS was FOCUS
 (2) Previous PROCESS was NOT ACTIVE
 (3) Next previous PROCESS was FOCUS
 (4) Next previous PROCESS was ACTIVE

THEN (1) Match the FOCUS OF ATTENTION rules

The previous meta-rules are essential for the proper functioning of the system, and are included in the LTM irrespective of the processing strategy. Other meta-rules are programmed to implement a specific control strategy. The following sequence of rules, for example, alternatively invoke region analysis and the focus of attention

META-RULE (9)

IF (1) Previous PROCESS was FOCUS
 (2) Previous PROCESS was ACTIVE

THEN (1) Match the REGION analysis rules

META-RULE (10)

IF (1) Previous PROCESS was REGIONS
 (2) Previous PROCESS was ACTIVE

THEN (1) Match the REGION analysis rules

META-RULE (11)

IF (1) Previous PROCESS was REGIONS
 (2) Previous PROCESS was NOT ACTIVE

THEN (1) Match the FOCUS OF ATTENTION rules

The effect of these is to successively match the region rules on all the regions in the current area of attention

Variations on the above are provided by the multitude of cases that occur when the different processes are interchanged. A particular strategy can be designed and translated into a flow chart, similar to the one in figure (5 14). The chart can then be used to select the subset of meta-rules that satisfy its requirements. Meta-rules (1) to (11) implement the strategy of figure (5 13). Each region is successively modified by all the rules before moving to other regions. Replacing the crossed line by the dotted line in figure (5 13) results in an alternative strategy. In this case, the system moves to another region after a single modification of the current region by the first rule to fire. This is accomplished by replacing Meta-Rule (10) by

META-RULE (12)

IF (1) Previous PROCESS was REGIONS
 (2) Previous PROCESS was ACTIVE

THEN (1) Match the FOCUS OF ATTENTION rules

The elements that constitute the system processing strategy and the choice of a particular strategy will be discussed in detail in chapter 6. Other examples of combinations of meta-rules that execute different strategies will be given in section 6 2 3. A complete listing of the focus of attention rules and the meta-rules is given in Appendix B.

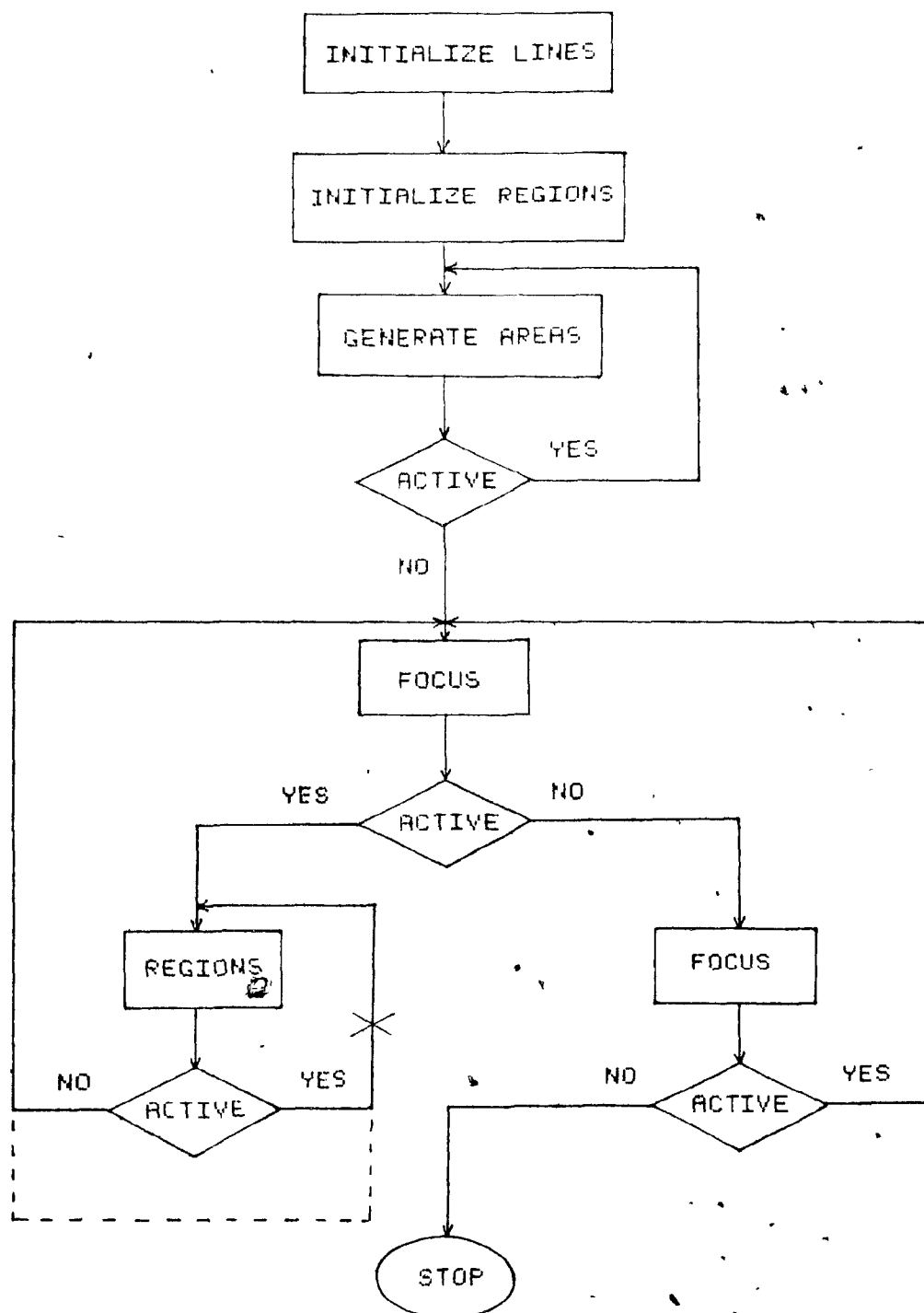


Figure (5 14) The flow chart of a system strategy

In this chapter, we have described the knowledge and control rules. The structure of the rules was analysed. A method for symbolically representing the knowledge was introduced. The conditions and actions that define our "segmentation language" were listed. The three types of knowledge rules were reviewed, showing how various segmentation heuristics are coded. Finally, the role of the focus of attention rules and the meta-rules in designing and implementing system control strategies was described.

The rules are seen to be modular, independent and symbolic. The importance of these factors for rule-based systems are self-evident. For computer vision, knowledge is complex and diverse, which renders such properties vital. The difficulties in designing the control of the rule-based vision system are only balanced by the use of the powerful, yet flexible, methodology embedded in the control rules.

CHAPTER 6

THE SYSTEM STRATEGY

6.1. INTRODUCTION

Setting strategy is an important factor in the design of a system, since it bears a direct consequence on how it will function. In the case of a rule-based system, this involves the fundamental issue of determining the order in which the rules will be matched against the data. In addition, for systems that deal with complex data structures, as is the case with a computer vision system, an ordering must be imposed on the data to be matched. For the rule-based image segmentation system described here, both data ordering and rule ordering are part of the system strategy.

The data in the image are composed of areas that contain the regions and lines that define a segmentation. A rule cycle begins by matching the conditions of a particular rule to specific data entries. These conditions require that the features of a region, line, or area, be tested for the

presence of a certain data configuration. Therefore, each KNOWLEDGE MODULE must have selected for it the specific region, line, and area to be considered at any point in the processing. Conceptually, the system is searching the image for the occurrence of the data configuration represented by a rule. It is the order of that search that must be established here. This translates into the problem of selecting the next area of attention to process, and specifying the order, within this area, in which the regions and lines are to be visited.

Image processing systems in the past have not seriously addressed the problem of determining the order in which an image is to be processed. However, there is ample proof that such an ordering affects the results of the analysis. This is evident, for example, in region growing systems, in which initial regions are merged in a pre-specified order to form larger regions. This data ordering was shown experimentally to affect the quality of the output [Nazif and Levine, 1981]. This is true irrespective of the fact that the merging criteria remain unchanged. Thus, this aspect of setting strategy is an important one, and in fact, it is responsible for a lot of the errors in the results of segmenting natural scenes using existing segmentation systems.

We have studied the problem of selecting the order in which a low level system should process an image. The inclusion of focus of attention areas in the data structure is significant [Levine and Shaheen, 1981, Levine and

Nazif, 1982b] These areas guide the system to parts of the image that should be processed first. The FOCUS OF ATTENTION MODULE selects the appropriate area by using a decision-making process that is based on the properties of different areas. Once this is done, the SUPERVISOR determines the method of data selection within the area by assigning priorities to the FOCUS OF ATTENTION rules. Furthermore, this data selection strategy is not a fixed one. It varies from one area to the next, according to the properties of the data within each area. We thus introduce the notion of dynamic data selection for processing an image. The FOCUS OF ATTENTION MODULE will execute the strategy that is now embodied in its rules to choose specific regions and lines within the selected area.

In a more general sense, setting the strategy includes not only specifying the method of data selection, but also establishing priorities among the processing criteria embodied in the rules. Before any of the KNOWLEDGE MODULES can match its rule set to the data, it must first be selected. Obviously, this requires establishing a priority among the KNOWLEDGE MODULES. Furthermore, more than one rule within the same set of knowledge rules could be applicable at any time. When such conflicts arise, a priority ordering is needed to specify which one of a group of rules should be selected to execute its actions. The SUPERVISOR determines these priorities, as well as the priority among the KNOWLEDGE MODULES. Unlike previous rule-based systems, the conflict

resolution mechanism used is a dynamic one as well. The order in which the rules are matched can actually be varied, based on the characteristics of the data. Rule re-ordering has been used in other rule-based systems [Davis, 1980]. However, the system described here provides the additional step of dynamically adjusting the method by which the rules are ordered.

The introduction of a dynamic element to both aspects of strategy described above is possible mainly because of the creation of a set of segmentation measures for region and line processing. These are computed for each area, stored in the STM, and are thus available to all of the processing modules. They have the additional property of being dynamic, in the sense that they reflect the state of the segmentation at any point in time. These measures provide an implicit bottom-up feedback path, which influences strategy setting in each area. The SUPERVISOR uses this information to compute the priorities of the KNOWLEDGE MODULES, the priorities of the rules within each module, and the priorities of the FOCUS OF ATTENTION rules. In doing so, it defines both a conflict resolution strategy and a data selection strategy. These constitute the top-down control paths of the strategy setting loop.

In the next sections, we will define how the strategy for the rule-based segmentation system is determined. To do that we first define what the elements that compose a strategy are. We then formulate the problem of strategy

selection as that of a search within the space composed by these elements, for the most appropriate strategy to use for each data configuration. To solve this dynamic search problem, we define performance parameters that are evaluated by the system at different parts of the image. They are used to determine the elements of strategy for each of these parts individually. We will also introduce the constraints that define the relation between the strategy elements and each of the performance parameters.

The framework for the solution is shown to be that of a decision-making process, in which decision functions are evaluated from the performance parameters, and are then used to instantiate the system strategy elements. The concepts of fuzzy set theory serve as the design tool for this process. Fuzzy logic will be used in representing the constraints, defining the objectives, and designing the decision-making algorithm.

6.2. STRATEGY DEFINITION

In order to define a strategy for the rule-based segmentation system, we need to answer two basic questions:

(1) Where (in the image) do we go to next?

(2) What do we do when we get there?

The answer to the first question provides the means for selecting the next data entry in the STM to match the rules

on. Within the context of the low level processing system described here, two levels of data selection are indicated. A given strategy should first detail the global method by which the next area in the image is to be selected for processing. It should then proceed to specify the criteria for locally selecting the next region and the next line in that area.

The answer to the second question also constitutes part of the local strategy. After the next data entry to match on has been selected, be it a region or a line, there still remains more than one alternative. These correspond to the different options provided by having more than one knowledge source to instantiate, or in other words, more than one rule to match. Would it be better to match REGION or LINE rules first? Do we give priority to merging over splitting or should we do the opposite? For lines, do we first add new lines or delete existing ones? When do we move on to the next region or line? In general, given a certain data configuration, more than one rule can match, and the question is if this happens, which rule should fire? For rule-based systems this is known as the conflict resolution problem. The local strategy within an area should also provide the solution to this problem by specifying an ordering on the rules to be matched. This ordering will thus vary from one area to another, depending on the features that describe each area.

To summarize, the strategy of our rule-based segmentation system is a twofold one. First, a bi-level data selection process comprising the global task of selecting the next area in the image, and the local method of moving among the different regions and lines within each area. Second, a strategy for locally adjusting the priority of rules in the model to best accommodate the properties of each area. Area selection is performed by the FOCUS OF ATTENTION module through a dynamic data driven decision making procedure, that will be described in section 6.4.2. We will now concentrate on defining the strategy within each area more precisely by analysing its two constituents: data selection and rule priority. Each will be decomposed into basic elements whose union will define a strategy vector for each area. In all, six components will be introduced and will be referred to as the "elements" of strategy. These are selected to be independent entities, that completely address the strategic aspects of the problem. Each element can assume specific "states" that are attached to it by the system.

6.2.1. Data Selection Strategy

Data selection contributes three elements to the strategy vector. Following is a discussion of each.

6.2.1.1. Region Path Strategy

This strategy element determines which region to address next after processing the current region. The FOCUS OF ATTENTION MODULE is the one responsible for obtaining the next region. It does so by matching its own rules against the data in the STM. It is the actions of these rules that determine the method by which the next region is to be fetched. The different actions provided by the FOCUS OF ATTENTION MODULE have been described in section 4.5.2. These actions correspond to the different states of the region path strategy. They included selecting the next region by adjacency values, by size, or by position. A specification of the region path strategy is thus equivalent to the selection of one of the above actions.

6.2.1.2. Line Path Strategy

This strategy element specifies how the next line is to be selected. In section 4.5.3, the actions that can be invoked by the FOCUS OF ATTENTION MODULE were described. They included fetching the next line based on its distance from the current line, its length, or its average gradient. The line path strategy has states that correspond to each of these actions.

6.2.1.3. Updating Strategy

When a region is visited by the system, the REGION ANALYZER tests its rules on the features of this region. If all the conditions of a rule match, the rule will fire, and an action will be executed. This will result in the modification of the current region. Once this is done, the SUPERVISOR has two recourses. It may again test the REGION ANALYZER rules to the same region, so that the same rule or others, may fire and further modify this region. This can be repeated until no more rules match their conditions on the features of the current region. Alternatively, the SUPERVISOR can invoke the FOCUS OF ATTENTION MODULE to fetch a new region. In this case, processing moves to the next region, so that rules will not modify the same region twice until all the other regions in the same area have been visited at least once each.

The updating strategy for regions has two states that correspond to the two alternatives described above. They are termed "successive" and "single" updating, respectively. In the first case, each region is "successively" updated to the maximum possible extent, according to the knowledge embedded in all applicable rules. This renders it highly unlikely to be further modified in the future. As soon as all the regions in an area are visited, most of the possible processing will have been accomplished. A second pass will only yield minimal results in terms of changing the final output.

This is not the case, however, when "single" updating is used. Whenever a region is altered by a "single" action, the system will move on to another region, usually in the same proximity, where the effect of the first change might induce an action on the new region. This will continue until all the regions have been inspected, at which point a return to a previously visited region becomes possible. A larger number of passes will be required to ensure the application of all processing criteria to all regions. Single updating is more conservative because it allows incremental changes to propagate through the area. Successive updating, on the other hand, is computationally more efficient. This is due to the fact that for some regions the matching of all rules does not produce a result, nevertheless these regions will be visited more than once during single updating as opposed to only once in successive updating. The trade-off is between computational efficiency and susceptibility to local errors. It is worthwhile to note that although the above reasoning has used regions as an example, it is equally applicable to lines as well. We now turn our attention to the second aspect of strategy.

6.2.2. Rule Priority

As with all rule-based systems, the order in which the rules are matched constitutes an important part of the system design. This affects both its behaviour and its performance. With a rigid design, rule ordering is embedded in the

structure of the system itself. A change in that ordering is virtually impossible without major modifications. This is not the case in production systems, where the production rules are kept separate from the processing modules of the systems. Therefore, an explicit rule ordering must be specified in order to carry out the rule matching process. This is usually done by attaching priorities to the rules, so that in case of conflict, the rule with higher priority will fire. This priority can be explicit, or it can be implicit by the temporal order in which the rules are matched. In both cases, the ordering specified is static and cannot be changed during processing. In fact, all rule-based systems developed so far using the production system approach have static conflict resolution methodologies.

The system described here implements a dynamic strategy, which implies a dynamic change in the method by which the rules are ordered. The next three strategy elements define priorities among rule sets, as well as within each

6.2.2.1. Processor Priority

One of the questions that arises during processing is whether to process regions or lines. The system is faced with this choice at the end of every rule cycle. The selection is accomplished by the SUPERVISOR whose meta-rules can specify one of two actions: match REGION rules, or match LINE rules. This strategy element establishes priorities for the rule sets by specifying whether first to match REGION or

LINE rules

We now carry the above argument one step further, in order to establish priorities within each of the rule sets. Two strategy elements will be distinguished at that level, they are REGION rule priority, and LINE rule priority.

6.2.2.2. Region Rule Priority

One way to rank the rules is by their actions. An ordering is imposed so that rules with certain actions have priority over rules with others. For REGION rules, two types of actions are possible, merging and splitting. Depending on the performance parameters that describe a certain area, the system may impose a priority for merging over splitting in that area, or vice versa. A dynamic ranking of the rules by their actions is thus established.

A further ordering of rules with the same type of action is possible. This is done by using the conditions associated with the rules to specify priorities for rules having the same action. One may argue that rules with larger number of conditions should have priority over those with a smaller number. In the system described here, conditions that take into account both region and line information take priority over those that consider only one or the other. Unlike the action-based priority, this condition-based ordering is a static one, that can be reflected in the order in which the rules are stored.

Rules with the same type of action are sorted according to their established priorities before the processing begins. The priority of the actions themselves remain to be established dynamically according to the data. This dynamic strategy element can have one of two states, namely, merging or splitting, depending on which action will have the higher priority setting.

6.2.2.3. Line Rule Priority

The argument described above for the REGION rules also applies here. A dynamic ordering is also imposed by the actions of the rules, and a static ordering is defined by the conditions of rules with the same type of action. The types of actions used are, of course, different. Again, two classes are distinguishable, those that add new lines, and those that delete existing ones. The former includes actions that insert, extend, and join lines, while the latter includes those that delete a line and merge two lines into one. The two states of this element of dynamic strategy will thus be referred to as the add and delete states.

This concludes our discussion of the various elements of the local strategy of the system. Table (6.1) summarizes the strategy elements that constitute the state vector for each area and shows their possible states. The next section explains how the system dynamically modifies its processing strategy to reflect the particular states given to each of the strategy elements described above.

STRATEGY ELEMENT -----	STATE -----
Region Path Strategy	highest adjacency lowest adjacency largest adjacent smallest adjacent raster scan higher label
Line Path Strategy	closest in front closest behind closest parallel longest near shortest near strongest near weakest near raster scan higher label
Updating Strategy	successive single
Processor Priority	regions lines
Region Rule Priority	merge split
Line Rule Priority	add delete

Table (6 1) The dynamic elements of strategy
and their states

6.2.3. Dynamic Strategy Setting

The local strategy changes whenever a new area in the image is brought to the attention of the system by the FOCUS OF ATTENTION MODULE. Based on the properties of this area, reflected by its performance parameters, the SUPERVISOR will conduct a decision-making process which assigns a particular state to each of the strategy elements, as will be discussed in section 6 4. Meanwhile, this section describes how the SUPERVISOR alters the processing strategy to reflect the new

states

Let us first examine how each of the strategy elements is set by the system. As discussed previously, the region path strategy and the line path strategy are executed by the FOCUS OF ATTENTION MODULE. Each state associated with these strategies has a corresponding rule action that replaces the current region or line by the next one. To change the path strategy for regions or lines the SUPERVISOR has to change the action of a FOCUS OF ATTENTION rule, or equivalently replace the rule by another that has a new action.

Successive and single updating are set up by different combinations of meta-rules. In general, the KNOWLEDGE MODULES will match their rules on the current region and the current line. Therefore, they will keep doing so until the FOCUS OF ATTENTION module is invoked to bring forth a new region or line. Successive updating is thus accomplished by recursively invoking the REGION or LINE ANALYZERS until no rules can match, wherein the FOCUS OF ATTENTION MODULE must be invoked. If, on the other hand, a new region (or line) is brought in after each successful match and modification of the current region (or line), a single updating strategy would result. This can be accomplished by alternating the REGION and LINE ANALYZERS, on the one hand, followed by the FOCUS OF ATTENTION MODULE. Consider the effect of recursively matching the following two meta-rules in a cyclic manner.

META-RULE (10)

IF (1) Previous PROCESS was REGIONS
 (2) Previous PROCESS was ACTIVE

THEN (1) Match the REGION analysis rules

META-RULE (11)

IF (1) Previous PROCESS was REGIONS
 (2) Previous PROCESS was NOT ACTIVE

THEN (1) Match the FOCUS OF ATTENTION rules

Meta-rule (10) will match the REGION rules on the current region. If a match occurs, the region is modified, and the next rule cycle starts. The previous process will now be REGIONS, and it will have been ACTIVE because of the match in the previous cycle. Accordingly, meta-rule (11) will not match, whereas meta-rule (10) will fire again. This will continue to happen, until all the REGION rules fail to match on the current region, at which point the previous process (REGIONS) will be labelled NOT ACTIVE. Meta-rule (11) will fire and the FOCUS OF ATTENTION rules will be invoked to bring in a new region. The two rules are thus seen to implement a successive updating strategy.

On the other hand, if these rules are replaced by

META-RULE (9)

IF (1) Previous PROCESS was FOCUS
 (2) Previous PROCESS was ACTIVE

THEN (1) Match the REGION analysis rules.

META-RULE (13)

IF (1) Previous PROCESS was NOT FOCUS
 THEN (1) Match the FOCUS OF ATTENTION rules

a single updating strategy would result. This is because meta-rule (13) will invoke the FOCUS OF ATTENTION MODULE to bring in a new region, irrespective of whether any of the region rules in the previous cycle matched on the then current region. Meta-rule (9) will match the REGION rules only if a new region is brought in by the FOCUS OF ATTENTION MODULE. The implication here is that if successive updating is invoked, the pair of meta-rules (10) and (11) would be used in the order described. The same applies for single updating with meta-rules (9) and (13).

Process priority is also established by changing the meta-rules. The following set of meta-rules establishes a single updating strategy in which REGION rules have priority over LINE rules.

META-RULE (9)

IF (1) Previous PROCESS was FOCUS
 (2) Previous PROCESS was ACTIVE
 THEN (1) Match the REGION analysis rules.

META-RULE (14)

IF (1) Previous PROCESS was REGIONS
 THEN (1) Match the LINE analysis rules.

META-RULE (15)

 IF (1) Previous PROCESS was LINES
 THEN (1) Match FOCUS OF ATTENTION rules

Interchanging REGIONS and LINES above will reverse the process priorities

The REGION and LINE rule priorities are established by assigning higher priorities to rules that merge or split regions and rules that add or delete lines, depending on the states of the respective strategy elements. The KNOWLEDGE MODULES will always match rules with higher priorities first. Thus, when all the conditions of a rule match, the rule can fire, knowing that all the rules that have not been matched are of equal or lower priority than this rule.

We can conclude from the above analysis that there are three independent steps involved in strategy setting:

- (1) Changing the actions of FOCUS OF ATTENTION rules, or equivalently, replacing rules by others that have the same conditions but different actions
- (2) Replacing meta-rules that implement one strategy, by others that implement a different strategy
- (3) Changing the priority of REGION and LINE knowledge rules

In order to replace a rule by another, we include both the old and new rules in the model. We then define a negative priority of -1, which indicates to the processing

modules that a rule should be ignored. In changing the strategy, the SUPERVISOR can now substitute two rules by assigning a priority of -1 to the rule to be replaced and changing the priority of the new rule from -1 to a positive value. Applying this method ensures consistency in implementing the three steps cited above. The first two require rule replacement, which is now equivalent to changing the priority of rules, as is the case with the third step.

All the rules in the model are thus assigned priorities that reflect an initial strategy for the system. Meta-rules and FOCUS OF ATTENTION rules are included that represent all possible strategy states. Rules that are not part of the initial strategy are assigned negative priorities and are therefore ignored by the system. The SUPERVISOR will dynamically update the strategies by changing the priorities of knowledge rules, FOCUS OF ATTENTION rules, and meta-rules. This includes the possibility of changing some priorities from negative to positive and vice versa, and consequently bringing rules in and out of the system model.

The SUPERVISOR updates the system strategy based on the characteristics of the current area of attention. These are reflected by its performance parameters, which are the subject of discussion in the next section.

6.3. PERFORMANCE PARAMETERS

Measuring performance is a key factor in providing a feedback path by which a system can modify its strategy during processing. For systems with well defined goals, performance parameters can be found that measure the distance from the goal at any point in time. These parameters can then be used to optimize the path towards that goal. Unfortunately, image segmentation systems do not fall into this category, at least not at the low level of processing. A goal can only be defined precisely in high level terms. Segmentation of the image into regions that correspond to objects in the scene, or the isolation of a particular object from the background are examples of these.

In terms of low level image segmentation, we can only hope for a fuzzy definition of the goal. The primary reference used to evaluate the final result is that of human performance in segmenting the image. We can define a segmentation as a partition of the image into regions that are uniform among themselves, and that bear contrast to their adjacent neighbors. Uniformity and contrast are measured in terms of a set of low level features that can be evaluated over the image. For example, these features can be the average grey level intensities of the regions in black and white or color spaces. Two factors are not accounted for by this definition, although they do affect human evaluation: the presence of texture in an image and the existence of lines as independent separate entities. A third important

aspect is context which is assumed to function at a higher level.

The problem, then, is to find a set of measures that can be computed for a segmentation of an image, and that closely simulate the judgement rendered by a human evaluating the same segmentation. The definition given above provides a clue to these measures. What is needed are estimates of uniformity within regions, contrast across regions, as well as provision for lines and texture.

In this section, we introduce the optimization criteria that are involved in measuring the performance of image segmentation systems. These are intended as general purpose parameters that can be used (all or the appropriate subset) in evaluating, as well as guiding, the processing of any partitioning algorithm. They apply to region-based, line-based, and texture-based approaches to segmentation. In particular, the set of measures detailed here were designed to provide the means for setting the processing strategy of our rule-based segmentation system.

6.3.1. Region Uniformity

The uniformity of a feature over a region is inversely proportional to the variance of the values of that feature evaluated at every pixel belonging to that region. A nil value for the variance requires all pixels to have the exact same value for the feature, while a large variance would

indicate a large spread from the mean of the feature value across the region. Let the value of feature F at each pixel i belonging to region R_j be f_i , then the mean of these values $\bar{f}_j$ is given by

$$\bar{f}_j = \frac{\sum_{i \in R_j} f_i}{A_j} \quad (6.1)$$

where A_j is the area of region R_j . The variance of feature F over region R_j is then given by

$$\sigma_j^2 = \frac{\sum_{i \in R_j} (f_i - \bar{f}_j)^2}{A_j} \quad (6.2)$$

The uniformity measure of an area α is defined as

$$U_\alpha = 1 - \frac{\sum_{R_j \in \alpha} w_j \sigma_j^2}{N} \quad (6.3)$$

where w_j is the weight associated with the contribution of region R_j to the measure, and the summation is taken over all regions in the given area. N is a normalization factor designed to limit the maximum value of the measure to one:

$$N = \left(\sum_{R_j} w_j \right) \left(f_{\max} - f_{\min} \right)^2 / 2 \quad (6.4)$$

where $f_{\max}$ and $f_{\min}$ are the maximum and minimum feature

values over the area, respectively. This corresponds to the worst case condition for the measure when all regions have the maximum variance

$$\sigma_{\max}^2 = \frac{(f_{\max} - f_{\min})^2}{2} \quad (6.5)$$

Since uniformity is basically a property that is increasingly noticeable with size, the contribution of each region to the uniformity of the whole area is taken to be proportional to the size of that region. The weight w_j for region R_j will thus be equal to the area of the region A_j . Substituting equations (6.2) and (6.4) in equation (6.3), we obtain the following for the region uniformity measure of an area:

$$U_{\alpha} = \frac{2 \sum_{R \in \alpha} \sum_{i \in R} (f_i - \bar{f}_j)^2}{\left(\sum_{R \in \alpha} A_j \right) (f_{\max} - f_{\min})^2} \quad (6.6)$$

The measure is seen to be a global one that represents all the regions in the area. However, each region's contribution to the measure depends only on the features of that particular region, and can, in fact, be computed separately. We can write

$$U_{\alpha} = 1 - \sum_{R \in \alpha} u_{j\alpha} \quad (6.7)$$

where $u_{j\alpha}$ is the contribution of region R_j to the measure.

given by

$$u_{j\alpha} = \frac{2 \sum_{i \in R_j} (f_i - \bar{f}_j)^2}{\left(\sum_{k \in \alpha} A_k \right) (f_{\max} - f_{\min})^2} \quad (6.8)$$

This can be simply written as

$$u_{j\alpha} = \frac{A_j \sigma_j^2}{A_\alpha \sigma_{\max}^2} \quad (6.9)$$

where A_α is the sum of the areas of the regions in area α .

This renders the measure easy to update whenever regions change due to splitting or merging. The efficiency of the computation is evident, in that we need only recompute the contributions of the regions that undergo a change; the contributions of other regions remain constant. In case of merging two regions, for example, the contribution of each region before merging is first subtracted from the measure, and the contribution of the resulting merged region is then added. This is an important property, since the measure has to be updated after every modification of the regions.

6.3.2. Region Contrast

The previous measure takes into account the nonuniformity inside each region. Contrast can thus be computed on the average values of the features of adjacent regions. The assumption here is one of independence. Uniformity is computed as a measure within each region independent of the surrounding regions, while contrast is computed between adjacent regions assuming that they have a uniform feature value equal to the average of that feature. The contrast between two regions R_i and R_j is thus given by

$$c_{ij} = \frac{|\bar{f}_i - \bar{f}_j|}{\bar{f}_i + \bar{f}_j} = c_{ji} \quad (6.10)$$

This is seen to have a minimum value of zero and a maximum value of one. The contrast measure for a region is equal to a weighted sum of the contrast of that region with all of its adjacent neighbors. Each neighboring region R_i contributes to the summation with a value that is proportional to the adjacency between it and the region R_j for which the measure is computed. Let this adjacency value be a_{ji} , the measure for region R_j is then given by

$$c_j = \sum_{i \text{ adj } R_j} a_{ji} c_{ij} \quad (6.11)$$

Note that the summation of the adjacency values a_{ji} for all regions R_i adjacent to R_j is equal to one. The bounds of the measure thus remain unchanged.

In order to obtain a single contrast measure for the whole area, we need to compute a weighted sum of the contributions of each region in that area. In general, the contrast measure for area α is given by

$$C_{\alpha} = \frac{\sum_{R \in \alpha} \sum_j w_j c_j}{\sum_{R \in \alpha} \sum_j w_j} \quad (6.12)$$

where w_j is the weight assigned to region R_j .

As in the case of the uniformity measure, the weight assigned to the contribution of each region to the contrast measure depends on its size. The manner by which the weights vary with size, however, is not a linear one. Since the goal of these measures is to "imitate" human judgement, let us consider how the human visual system deals with contrast. Experiments that study the effect of spatial frequency on our perception of contrast yield what is known as contrast sensitivity curves [Campbell and Robson, 1968]. These are plots of the lowest contrast value between stripes of a certain spatial frequency, that enable humans to distinguish between these stripes, as a function of the spatial frequency. An example of such a plot for a fixed average intensity level is shown in figure (6.1). The sensitivity decreases for very high and very low spatial frequencies and has an optimal value at a frequency that corresponds to approximately $3'$ of visual angle.

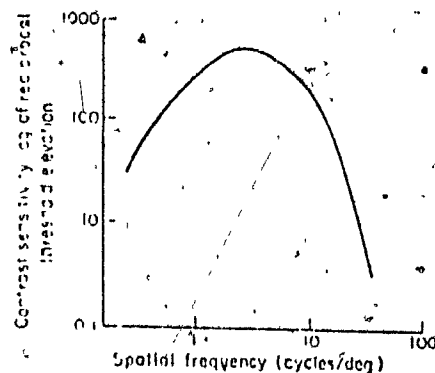


Figure (6.1) : Example of contrast sensitivity curve.

From [Campbell and Robson, 1968].

Applying the above observations to the contrast between regions in a segmented image, we can argue that there is an optimal region size for which contrast between regions visually affects the results of segmentation the most. This is because, as the spatial frequency increases (the size of the region decreases), the perception of contrast is enhanced until we reach the optimal size. As the size of regions decreases beyond that, contrast perception deteriorates because of the limitations imposed by visual acuity. The curve shown in figure (6.2) represents the weights given to regions as a function of region area, in the form of a normal distribution.

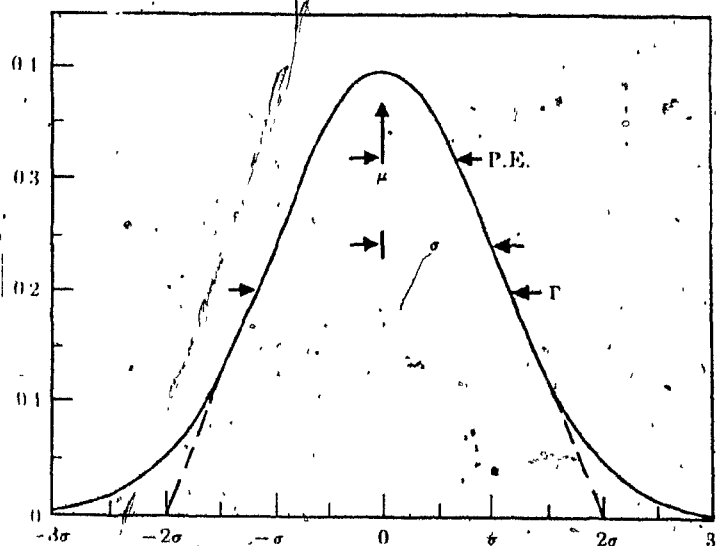


Figure (6.2) : Weights as a function of region size.

This was chosen as a crude, but simple approximation of a single valued function that simulates the behaviour described above. The mean of the curve corresponds to the optimal region size, and the standard deviation is chosen so that the weights will approach zero at the limits of visual acuity (this corresponds to the image resolution in computer vision). The weight for region R_j is thus given by

$$w_j = \frac{1}{\sqrt{2\pi}\sigma} \exp - \left[\frac{(R_j - \mu)^2}{\sigma^2} \right] \quad (6.13)$$

where μ and σ are the mean and standard deviation of the curve, respectively.

The previous two measures account for the segmentation of an image into regions that are uniform and distinct. They are seen to be complementary. Both measures must be high to indicate a good segmentation. A composite measure of segmentation must take this into account. In the following two sections, we discuss the effect of lines on the evaluation of a segmentation, by presenting two appropriate measures.

6.3.3. Line Contrast

This measure is designed to check the validity of existing lines in an image segmentation. We infer the presence of a line in an image if one of two things occur:

- (1) There exists an appropriate amount of contrast in the features of regions on either side of the line.
- (2) There is enough evidence, in terms of a local gradient in the features of pixels belonging to the line, to justify its presence.

The first condition accounts for most of the lines that humans infer in an image. The second condition is usually satisfied when the first one is. It is included separately to account for certain special cases where lines exist as separate entities, that are independent of the regions that surround them. An example of this occurs when a thin object, whose width is the same as that of a line, occludes a uniform object in the background. Since the regions on both side of the line belong to the same object, there is virtually no

contrast between them. Still there is a strong local gradient brought about by the existence of the thin object.

The measure defined here is a composite one, in order to account for both types of lines cited above. The first category is best represented by the contrast between regions on both sides of a line. On the other hand, a local gradient measure will be more useful for lines belonging to the second category. These contribute weakly to a contrast measure even when they are rightfully in place. Let the contrast of a feature F across line L_j be d_j ; then

$$d_j = \frac{|F_L - F_R|}{F_L + F_R} \quad (6.14)$$

where

$$F_L = \sum_i v_{il} \bar{f}_i, \quad F_R = \sum_k v_{kr} \bar{f}_k \quad (6.15)$$

and v_{il} is the adjacency value of a region R_i to the left of the line, $\bar{f}_i$ is the average value of feature F over that region, v_{kr} is the adjacency of a region R_k to the right of the line, $\bar{f}_k$ is the average feature value over that region, and the summations are taken over all regions to the left and right of the line, respectively.

For the second category, we define a measure of the local gradient across line L_j as g_j , where

$$g_j = \frac{G_j}{(f_{\max} - f_{\min})} \quad (6.16)$$

G_j is the average gradient of feature F over the pixels belonging to line L_j . $f_{\max}$ and $f_{\min}$ are the maximum and minimum feature values in the area as defined in section 6.3.1. The contrast measure given in equation (6.14) accounts for the first category of lines, which includes most of the lines in an image. The second group of lines is more adequately represented by the local gradient measure of equation (6.16). The latter are characterized by a high value of g_j coupled with a low value of d_j . A hybrid measure h_j can thus be constructed for the contrast across a line L_j :

$$h_j = \begin{cases} g_j & \text{if } c_j < \zeta \text{ and } g_j > 3 c_j \\ c_j & \text{otherwise} \end{cases} \quad (6.17)$$

and ζ is a constant near zero that indicates very low contrast. A line will thus contribute to the contrast measure through the properties of the regions on both sides of it. If, however, the region contrast is low, the local gradient across the line will be more representative of its contribution.

A composite measure of line contrast over an area α can now be computed from the individual line measures as follows:

$$H_{\alpha} = \frac{\sum_j L_j \alpha_j w_j h_j}{\sum_j L_j \alpha_j w_j} \quad (6.18)$$

where w_j is the weight assigned to the contribution of line L_j , and is taken to be equal to the length of the line. The summation is over all lines in the area.

6.3.4. Line Connectivity

The previous measure only accounts for the validity of lines present in the image segmentation. A complete measure of segmentation should also involve lines that should have been present, but are in fact missing. This kind of error in segmentation is more difficult to quantify, although easily detected by a human observer. If a process is able to observe the absence of lines, it would not have missed them in the first place. Nevertheless, some inference can be made about missing lines. A good clue to this is the presence of incomplete lines. By their nature, lines in an image tend to close on themselves, or on one another. This is also in accordance with one of the principles of Gestalt psychology, that of good continuity [Koffka, 1963]. Lines rarely stop suddenly, and disappear, or have an open end that goes nowhere.

Based on this argument, the measure developed here is one of connectivity of lines. The end of a line is said to be closed if it terminates on another line, or if it closes on itself. A line is closed if both its ends are closed. The number of lines that have one or both ends open is an indication of the degree of discontinuity in the lines. Let t_j be the number of closed ends of line L_j and l_j be the length of that line. Then,

$$T_{\alpha} = \frac{\sum_{L_j \in \alpha} \frac{t_j}{l_j}}{2 \sum_{L_j \in \alpha} \frac{1}{l_j}} \quad (6.19)$$

will represent a global measure of the connectivity of all the lines in area α . Note that t_j is a tri-valued quantity $\{0, 1, 2\}$, and hence the measure T_{α} is bounded between the values of zero and one. A low value of T_{α} would indicate that there is a large amount of discontinuity in the lines, because of the large number that have their ends open.

Although the measure described here does not account for all the lines missing from an image segmentation, it does go a long way in doing so. This is because lines in an image tend to be partially present, rather than totally missing. It is those totally missing lines that are not represented in the measure. Again the connectivity measure complements the contrast measure for lines, as is the case with the uniformity and contrast measures for regions.

In order to complete our evaluation of an image segmentation, other performance parameters are added to the above four measures, and are designed to account for the presence of texture in an area of the image.

6.3.5. Texture Measures

When we refer to region uniformity, we usually mean uniformity over grey level features. Thus uniform regions are those that do not contain any texture. However, a more general interpretation of uniformity would include texture features as well. If we extend this to the definition of segmentation given at the beginning of this section, the regions in an output segmentation would be required to be uniform in texture features. This means that textured regions would not be divided into the grey level uniform sub-regions representing the units of the textural pattern.

Instead, they would be grouped into single entities, despite the fact that their grey level uniformity would be quite low.

One of the main problems with this approach is that textural features are inherently global features that must be computed over an area in the image. Some approaches to texture analysis assign local values to these features at every pixel in the image [Levine, 1980]. This is done by computing the global measure over an area surrounding each pixel. The choice of that area is what has become known as the window size problem [Riseman and Arbib, 1977]. A large window is likely to include a range of values at the boundaries between

textured regions. A small window, on the other hand, will be more sensitive to local variations due to the individual patterns of a texture. Efforts to determine an optimal window size have had limited success [Rosenfeld and Thurston, 1971], and an exhaustive search would add to the computational burden.

Our approach to the problem is quite different. Regions are defined to be uniform over grey level features. The first definition of segmentation is thus applicable, and so are the measures described in sections 6.3.1 and 6.3.2. Texture areas are then defined explicitly, and are represented as a group of uniform grey level regions that correspond to the primitive elements of texture [Matsuyama et al., 1982]. These regions are usually small in size, and exist in clusters of relatively large numbers. Texture properties can now be computed by averaging the feature values over the regions. A histogram of the average intensity of all the regions belonging to the same area, for example, would be multi-modal. This is because each group of regions (e.g., background and foreground) will contribute a distinctive intensity value, and hence will produce a separate peak in the histogram. Thus, the elementary regions replace the windows that are normally required for computing average textural features, and local values at every pixel in the image are no longer necessary. This solves the window size problem. A global measure over the regions will indicate whether they indeed belong to the same texture. The

computational efficiency is also seen to improve drastically.

Each focus of attention area has a label that classifies it as being "textured", "smooth", or as being "bounded" by long connected lines. The four performance measures described above are computed for each area. It is evident that the relative importance of each measure is dependent on the type of area for which it is used. Region uniformity is less important in textured areas than in smooth ones. Connectivity is important for textured and bounded areas, contrast is important in smooth and textured areas, although more so for regions than lines. Thus, the preceding measures do not have the same weight in different areas. This suggests the inclusion of the area type Ψ_α , deduced by the system, as a performance parameter for area α . The latter can then be used to emphasize the appropriate subset of measures in a particular area.

Other important parameters for an area are the number of regions and the number of lines it contains. In addition to being clues for texture, as performance parameters they have a large influence on setting strategy. These numbers are normalized over the areas by computing the number of regions (and lines) per unit area for the whole image and comparing it with the same quantity for each area. Consider the value

$$R_\alpha = \frac{NR_\alpha / A_\alpha}{NR_T / A_T} = \frac{NR_\alpha \cdot A_T}{NR_T \cdot A_\alpha} \quad (6.20)$$

where NR_α and NR_T are the number of regions in area α and in the whole image, respectively. If R_α is greater than unity, the density of regions in α will be greater than the average density for the whole image, and vice versa. A value in the range $[0,1]$ can then be computed by dividing by a multiple of the average density (e.g., 3), and truncating any higher values. Similarly for lines, we have

$$L_\alpha = \frac{NL_\alpha / A_\alpha}{NL_T / A_T} = \frac{NL_\alpha \cdot A_T}{NL_T \cdot A_\alpha} \quad (6.21)$$

The performance vector for each area in the image will thus include components for region uniformity, region contrast, line contrast, line connectivity, number of regions, and number of lines. Note that the first three components are computed per feature in the image. For black and white images, each component can be computed for the grey level intensity of regions and lines. For color images, each measure will contribute three elements to the performance vector, one for each color channel intensity. In general, the uniformity and contrast of other features of regions and lines can be added to the performance vector. For m such features, the performance vector $\bar{P}_\alpha$ for area α will have $M = 3m+3$ components, and is given by

$$\bar{P}_\alpha = [\overset{1}{U_\alpha}, \dots, \overset{m}{U_\alpha}, \overset{1}{C_\alpha}, \dots, \overset{m}{C_\alpha}, \overset{1}{H_\alpha}, \dots, \overset{m}{H_\alpha}, T_\alpha, R_\alpha, L_\alpha] \quad (6.22)$$

where U_{α}^1 to U_{α}^m are m region uniformity measures for area α ,

C_{α}^1 to C_{α}^m are m region contrast measures for area α ,

H_{α}^1 to H_{α}^m are m line contrast measures for area α ,

T_{α} is the line connectivity measure for area α ,

R_{α} is a normalized value for the number of regions in α ,

and L_{α} is a normalized value for the number of lines in α .

Each area α will be represented in the strategy determination process by its performance vector $\bar{P}_{\alpha}$ and its type Ψ_{α} .

6.4. STRATEGY DETERMINATION

The strategy of the rule-based system is determined in two steps. First, a global strategy evaluation scheme selects the next area of attention in the image. Once this is done, the local strategy used by the system within the selected area must be computed. This section details the methodology for determining the system strategy at both levels. A solution for global strategy determination is presented in the form of a fuzzy decision-making process. The constraints that govern the choice of a particular local strategy, and define the global strategy for area selection will be described. The fact that these constraints are themselves fuzzy in nature, will be shown to be the reason

for using fuzzy logic, rather than other standard decision-making methodologies. The problem of evaluating the local strategy is posed in the form of a "black box" model with specific inputs, outputs, and guiding constraints. The same principles used for global strategy selection are also used to solve for the local strategy model. We start by reviewing the basic ideas behind fuzzy logic and fuzzy decision-making, as they relate to the material discussed here.

6.4.1. The Principles Of Fuzzy Logic

The concepts of fuzzy sets were developed by L. A. Zadeh [Zadeh, 1965], and have since acquired considerable importance, due to their usefulness in a wide variety of applications. In this section, we present a brief description of fuzzy set theory, and the aspects and notation that are of relevance to our specific needs in designing a fuzzy decision-maker. A more complete description of the fundamentals of fuzzy logic can be found in [Zadeh, 1973; Zadeh, 1976; Gaines, 1976], and a recent bibliography on the subject and related applications is given in [Gaines and Kohout, 1977].

Fuzzy logic was introduced as a substitute for the standard binary logic in cases where a precise quantitative description of a problem is not feasible. This may be due to the complexity of the concepts involved, or to a deficiency in our understanding of these concepts. A large number of

problems in different application areas fall into this category. Disease diagnosis in medicine, recession in economics, meaning in linguistics, insanity in law, and recipes for cooking, are just a few examples of concepts and algorithms that are better described and understood in the context of a fuzzy, rather than precise, environment.

Mathematically, the main difference between fuzzy logic and binary logic is in the introduction of "degrees of membership" for elements in the sets they belong to. Let U be the non-fuzzy set of elements

$$U = [u_1, u_2, \dots, u_n] \quad (6.23)$$

then,

$$A = \left[\frac{\mu_1}{u_1}, \frac{\mu_2}{u_2}, \dots, \frac{\mu_n}{u_n} \right] \quad (6.24)$$

is a fuzzy subset of U , where μ_i is the degree of membership of element u_i in the set A . The latter reflects our confidence in the fact that an element belongs to a specific set, or in other words, the degree by which that element satisfies the criteria imposed by the set. These confidences are generally taken to lie in the interval $[0,1]$, where 0 and 1 denote no membership and full membership, respectively. Following the notation in [Zadeh, 1976], equation (6.24) can be expressed as

$$A = \int_U \mu_A(u) / u \quad (6.25)$$

If A and B are fuzzy subsets of U , then their intersection $A \cap B$, and their union $A \cup B$, are also fuzzy subsets of U .

Thus,

$$A \cap B = \int_U \mu_A(u) \vee \mu_B(u) / u \quad (6.26)$$

$$A \cup B = \int_U \mu_A(u) \wedge \mu_B(u) / u \quad (6.27)$$

where $\vee$ and $\wedge$ denote min and max, respectively. The complement of A is given by

$$\bar{A} = \int_U (1 - \mu_A(u)) / u \quad (6.28)$$

It is interesting to note that the intersection, union and complement operations defined in equations (6.26), (6.27), and (6.28), are actually the fuzzy equivalents of the logical operations AND, OR and NOT, respectively.

One important consequence of fuzzy set theory is the introduction of hedges for linguistic variables [Zadeh, 1972]. These are qualifiers that modify a fuzzy variable, and by doing so, map it into a different qualitative range. If A is a fuzzy set representing such a variable, then "very A " can be taken as A^2 . In general, A^γ will serve to decrease the membership confidence of elements in a fuzzy set for $\gamma > 1$. It will do so more for members that are of originally lower confidence than for those with higher confidences. The opposite effect is seen to happen for $\gamma < 1$. Before proceeding any further, we will give an example to clarify the ideas presented so far.

Example 6.4.1

Let U be the set $[10, 20, \dots, 80]$, and A be a fuzzy subset of U that represents the linguistic variable "young", where

$$A = \left[\frac{1.}{10}, \frac{.9}{20}, \frac{.8}{30}, \frac{.7}{40}, \frac{.5}{50}, \frac{.3}{60}, \frac{.1}{70}, \frac{0.}{80} \right] \quad (6.29)$$

Then "very young" can be given by

$$A^2 = \left[\frac{1.}{10}, \frac{.81}{20}, \frac{.64}{30}, \frac{.49}{40}, \frac{.25}{50}, \frac{.09}{60}, \frac{.01}{70}, \frac{0.}{80} \right] \quad (6.30)$$

"not very young" is then given by

$$\overline{A}^2 = \left[\frac{0.}{10}, \frac{.19}{20}, \frac{.36}{30}, \frac{.51}{40}, \frac{.75}{50}, \frac{.91}{60}, \frac{.99}{70}, \frac{1.}{80} \right] \quad (6.31)$$

and "young and not very young" is given by

$$A \cap \overline{A}^2 = \left[\frac{0.}{10}, \frac{.19}{20}, \frac{.36}{30}, \frac{.51}{40}, \frac{.5}{50}, \frac{.3}{60}, \frac{.1}{70}, \frac{0.}{80} \right] \quad (6.32)$$

The decision-making process can now be formulated using the concepts described above. A decision here is defined as the process of selecting among a finite number of alternatives, according to a defined set of objectives. The alternative that satisfies "all" of the objectives is chosen as the output of the decision-maker. The function of the latter is thus to evaluate each of the alternatives, in order to determine whether or not it satisfies each of the required objectives. However, in a large number of cases, the process of determining whether a particular objective is satisfied, is not given by a simple yes-or-no answer. A "fuzzier" outcome is more likely. The decision-maker is able to determine "how well" each given alternative satisfies each of

the required objectives. It can then select the one that "best" satisfies the objectives. The following example will help clarify the underlying concepts.

Example 6.4.2.

Let $X = [x_1, x_2, x_3, x_4]$ be the set of brand names for a particular product. It is required to select the brand name that is economical, effective, and available. The three criteria were evaluated for all the candidates, and a number that indicates how well each brand satisfies each of the criteria was computed. The following fuzzy subsets of X are thus defined:

$$Y_1 = \left[\frac{.8}{x_1}, \frac{.4}{x_2}, \frac{.6}{x_3}, \frac{.7}{x_4} \right] \text{ for economy,} \quad (6.33)$$

$$Y_2 = \left[\frac{.3}{x_1}, \frac{.6}{x_2}, \frac{.2}{x_3}, \frac{.9}{x_4} \right] \text{ for effectiveness,} \quad (6.34)$$

$$\text{and } Y_3 = \left[\frac{.2}{x_1}, \frac{.8}{x_2}, \frac{.6}{x_3}, \frac{.3}{x_4} \right] \text{ for availability.} \quad (6.35)$$

Following the argument presented in [Bellman and Zadeh, 1970] and [Zadeh, 1973], the objectives above can be stated as Y_1 "and" Y_2 "and" Y_3 . This is known as the rule of implied conjunction. In terms of the fuzzy subsets of potential candidates, the decision becomes

$$D = Y_1 \cap Y_2 \cap Y_3, \quad (6.36)$$

which is also a fuzzy subset on the candidates. The membership values of D indicate how well each candidate satisfies the set of objectives as a whole. For the set of

candidates given in example 6.4.2, D is thus given by

$$D = \left[\frac{.2}{x_1}, \frac{.4}{x_2}, \frac{.2}{x_3}, \frac{.3}{x_4} \right] \quad (6.37)$$

which implies that candidate x_2 best satisfies the combined objectives. The fuzzy decision-making process can thus be summarized in the following two steps:

- (1) For each alternative, select the objective with the lowest membership value.
- (2) The alternative with the highest membership in the resulting fuzzy subset is selected as the optimal decision.

This is seen to be a maxi-min process.

To complete our discussion, we consider a variation on the above decision-making process, brought about when the given objectives are not of equal importance [Yager, 1977]. Each objective is assigned a number $\gamma > 0$, to indicate the relative importance of that objective. The general form of the decision function then becomes:

$$D = Y_1^{\gamma_1} \cap Y_2^{\gamma_2} \cap Y_3^{\gamma_3} \cap \dots \cap Y_N^{\gamma_N} \quad (6.38)$$

where N is the number of objectives. The higher the power to which the fuzzy set of an objective is raised, the more difficult it is to satisfy by the given alternatives. An alternative that is weak when evaluated for an important objective, will be made even weaker. Stronger alternatives will not be affected to the same degree. The effect of

raising an objective's fuzzy set to a power thus provides a means for scaling the importance of that objective relative to the others.

The selection of the global and local strategies for our rule-based system will be shown, in the following two sections, to fit conveniently into the format given by the methodology described above.

6.4.2. Global Strategy Evaluation

Before determining the local strategy within an area in the image, the area must be selected. The FOCUS OF ATTENTION module examines all available areas, and selects, based on their properties, the most appropriate one. A basic constraint to be applied here is that all areas in the image should be processed in an even-handed manner, or in other words, each area should get its fair share of processing. This is necessary to allow the system to use potentially useful low level information distributed over different parts of the image. Another constraint is to avoid locking the system into a particular area of the image, where it could waste precious computation time processing fine detail, that bears little impact on the overall quality of the output. This could happen while other areas in the image are in more need of processing, because of relatively higher segmentation errors within them.

A good global strategy would thus select the area that: (i) is in more need of further processing, and (ii) has been less recently visited by the system. An area's need for processing is reflected by its performance parameters. The first four parameters described in section 6.3 serve this purpose fully. The amount of additional region processing that an area requires is reflected in how low the region uniformity and contrast measures are. Low values of line connectivity and contrast measures indicate the need for adding or deleting more lines, respectively. The system should thus select the area with the lowest overall values for these four parameters.

A parameter that reflects temporal order, rather than performance, is required in order to represent the second constraint mentioned above. Consider the recency ratio for area α given by

$$R_{\alpha} = \frac{\text{Number of areas visited since } \alpha \text{ was}}{\text{Total number of areas in the image}} \quad (6.39)$$

A high value of R_{α} would indicate that the area has not been visited recently, and thus is an eligible candidate for processing.

Our global strategy, put in words, is to select the area of lower region uniformity and contrast, lower line connectivity and contrast, and of higher recency ratio. Although we can express each of these measures mathematically, the constraint over each of them is not so

well-defined. The format is a fuzzy one, since it only indicates that a measure should be low or high. Furthermore, the method by which the measures should be combined to have an effect on the final decision, yields another source of imprecision. We know that an area should satisfy as many of the constraints as possible, in the best possible way. In practical situations, each area will satisfy each of the constraints to some degree. Thus, a precise mathematical function of the measures could be formulated to produce a combined effect. However, such a formula, involving multiplicative or additive effects, would be too simplistic and too precise. This is because we only have a fuzzy notion of what to expect. Too fuzzy, in fact, to be represented by a single-valued mathematical function. The choice is thus better described by using a linguistic framework, as opposed to a multi-variable formula. This provides the motivation for using the concepts described in the previous section in making the decision.

Referring to the performance parameters of section 6.3, we notice that every measure indicates how well an area satisfies a particular objective. A region uniformity measure of 0.8, for an area, for example, indicates that we are 80% confident that the regions in the area are uniform. The same applies to the other region and line measures (note that they are all normalized to lie in the interval $[0,1]$). The confidences required by the fuzzy decision-maker can be derived directly from these measures. If the value of a

performance measure for an area is x , the amount of further processing required to improve that measure is proportional to $1 - x$. The recency ratio defined by equation (6.39) provides a confidence value that is proportional to an area's temporal eligibility.

For each area, $M = 3m + 2$ objectives are defined, where m is the number of features of regions and lines used for the evaluation, as introduced by equation (6.22), section 6.3. They correspond to all m region uniformity, m region contrast, m line contrast, one line connectivity, and one recency ratio measures. Let Φ be the set of N areas in the image:

$$\Phi = [\alpha_1, \alpha_2, \dots, \alpha_N], \quad (6.40)$$

and $Y_1, Y_2, \dots, Y_M$ be the set of M objectives. Accordingly, we have for each α_i ,

$$Y_k = \int_{\Phi} (1 - U_{\alpha_i}^k) / \alpha_i, \quad 0 < k \leq m, \quad (6.41-a)$$

$$Y_k = \int_{\Phi} (1 - C_{\alpha_i}^{k-m}) / \alpha_i, \quad m < k \leq 2m, \quad (6.41-b)$$

$$Y_k = \int_{\Phi} (1 - H_{\alpha_i}^{k-2m}) / \alpha_i, \quad 2m < k \leq 3m, \quad (6.41-c)$$

$$Y_k = \int_{\Phi} (1 - T_{\alpha_i}) / \alpha_i, \quad k = 3m + 1, \quad (6.41-d)$$

$$Y_k = \int_{\Phi} R_{\alpha_i} / \alpha_i, \quad k = 3m + 2, \quad (6.41-e)$$

where $U_{\alpha_i}^k$, $C_{\alpha_i}^k$, $H_{\alpha_i}^k$, and $T_{\alpha_i}^k$ are defined as per equation (6.22)

The decision-making process takes place by first computing the degree by which all areas satisfy each of the M objectives. For a particular objective k , this will correspond to finding the fuzzy subset Y_k over the set of areas Φ .

$$Y_k = \left[\frac{y_{1k}}{\alpha_1}, \frac{y_{2k}}{\alpha_2}, \dots, \frac{y_{Nk}}{\alpha_N} \right] \quad (6.42)$$

The y_{ik} are computed using equations (6.41-a) to (6.41-e), depending on the value of k . The fuzzy decision set D is then computed from the M sets of objectives, so that

$$D = Y_1 \cap Y_2 \cap \dots \cap Y_M \quad (6.43)$$

or

$$D = \left[\frac{d_1}{\alpha_1}, \frac{d_2}{\alpha_2}, \dots, \frac{d_N}{\alpha_N} \right] \quad (6.44)$$

where

$$d_i = \min_{k=1}^M [y_{ik}] \quad (6.45)$$

Following the optimal decision rule, the system will select area α_j as the next area to process, if

$$d_j = \max_{i=1}^N [d_i] \quad (6.46)$$

The maxi-min process of fuzzy decision-making is given by equations (6.45) and (6.46). The objective that is minimally

satisfied by an area is selected to represent it. The area with the highest value for the latter will constitute the output of the decision-maker.

To ensure equal participation by all performance criteria in this global process, the fuzzy sets of objectives are all raised to unity power. In order to give the recency ratio an edge over other objectives, its fuzzy set can be raised to the power 2. Equation (6.41-e) can be re-written as

$$Y_{3m+2} = \int_{\phi} (R'_{\alpha_i})^2 / \alpha_i \quad (6.47)$$

This means that the membership value for each area in this set will decrease, and more so for areas with lower value than for others. That objective will be more likely selected as the minimally satisfied for more areas, thus increasing its influence on the final decision.

6.4.3 Local Strategy Evaluation

Once the global strategy has selected an area in the image, it remains to define the local strategy within the area. In section 6.2, we described the elements of local strategy, their states, and how they can be adjusted by the system. Recall that at the local level, both data selection and rule priority are necessary to define this strategy. We have also introduced in section 6.3, performance parameters that can be evaluated for different areas in the image.

These parameters have been designed to measure the quality of a segmentation over an area, and thus provide the means for adjusting the local strategy within that area. They can do so by influencing the individual strategy elements into particular states that are more suitable for the data configuration in the area, as represented by these parameters. The following problem presents itself: Given a performance vector $\bar{P}_\alpha$ for area α , how can we determine the elements of the strategy vector $\bar{S}_\alpha$ for that area. What we are actually looking for is the model F represented by the "black box" in figure (6.3)

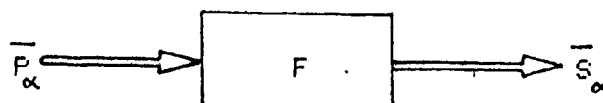


Figure (6.3) : Model for strategy evaluation

The constraints imposed on this model are brought about by the manner in which each performance parameter affects each strategy element. One can say that line analysis should have priority over region analysis if the region measures of uniformity and contrast are high, and the line measures of contrast and connectivity are low. This is an indication that the image needs more line processing than region processing. The opposite is also true, and variations on the above are possible. More splitting is needed at low region uniformity and more merging at low region contrast. Adding

lines is preferable for low connectivity areas, while deleting lines is more useful in smooth areas. A successive updating strategy is more efficient if the amount of processing left is low, as reflected by high values for the region and line measures. On the other hand, the more conservative single updating strategy is safer if the measures are low, and if the number of regions and lines are high. Bounded areas may invite region merging, while textured areas allow for more line deletion. The path strategy is also affected by these constraints. While low region uniformity suggests visiting larger regions first, low region contrast suggests moving through high adjacency values. Smaller size regions are more influential for texture areas, while the opposite is true for smooth areas.

These are just examples of the types of constraints present, and are not meant to constitute an exhaustive list. Clearly, not every performance parameter affects every strategy element, at least not to the same extent. Some constraints seem more important than others. Two factors affect the presence and relative importance of these constraints: improving the efficiency of the computation, and enhancing the quality of the output segmentation. In chapter 7, we will describe experiments that study the relation between the performance parameters and the various elements of strategy.

A consistent relationship can be obtained by generalizing the effects of performance parameters on strategy elements. This can be done by explicitly naming the effect of every parameter on all the states of all strategy elements. Two aspects can be coded: the direction of the effect, and its magnitude. A performance parameter can promote the instantiation of a particular state for one of the strategy elements in one of two different ways. The higher the value of the parameter, the more it would advocate the choice of that state. This will be termed a positive effect. The second and opposite effect, when a lower parameter value is more suggestive of the selection of the affected state, is termed the negative effect.

Table (6.2) represents the direction of the relation between performance parameters and strategy elements. The "+" and "-" signs correspond to the positive and negative effects, respectively. An entry is included whenever the direction of influence of a parameter on a state is known, irrespective of the magnitude of that influence. If no known effect exists, however, the entry in the table is left blank.

As indicated previously, the extent by which performance parameters affect strategy elements is not constant. Some parameters are more important than others for certain elements. The relative influence of the parameters on the elements will also depend on the type of area for which the strategy is evaluated. Table (6.3) summarizes this magnitude effect. The entries in the table code the influences into

Strategy	Performance Parameters					
Elements	U	C	H	T	R	L

High Adjacency	+	+			+	-
Low Adjacency	-	-			+	
Largest Size	+	-			+	
Smallest Size	-	+			-	

Nearest			-	-		-
Longest			+	-		-
Shortest			-	+		+
Strongest			+	-		-
Weakest			-	+		+

Successive	+	+	+	-	-	-
Single	-	-	-	+	+	+

Regions	-	-	+	+	+	-
Lines	+	+	-	-	-	+

Merge	+	-	-	-	+	-
Split	-	+	+	+	-	+

Add	-	+	+	-	+	-
Delete	+	-	-	+	-	+

Table (6.2) : Effect of performance parameters on the states of strategy elements.

Strategy Elements	Area Type	Performance Parameters					
		U	C	H	T	R	L
Region	smooth	VH	H	VL	VL	M	VL
Path	texture	H	VH	VL	VL	M	VL
Strategy	bounded	H	H	VL	VL	H	VL
Line	smooth	VL	VL	VH	H	VL	M
Path	texture	VL	VL	H	VH	VL	H
Strategy	bounded	VL	VL	M	M	VL	M
Updating	smooth	VH	H	H	M	M	M
Strategy	texture	H	VH	L	M	H	H
	bounded	H	H	M	M	H	M
Process	smooth	VH	H	H	M	VL	VL
Priority	texture	H	VH	H	VH	L	L
	bounded	H	H	M	M	L	VL
Region	smooth	VH	H	L	L	H	VL
Rules	texture	H	VH	L	L	VH	VL
Priority	bounded	H	H	L	L	VH	VL
Line	smooth	L	L	VH	H	VL	H
Rule	texture	L	L	H	VH	VL	VH
Priority	bounded	L	L	H	H	VL	H

Table (6.3) : Codes that reflect the relative importance of performance parameters in determining the various strategy states in different areas.

one of five categories: very low (VL), low (L), medium (M), high (H), and very high (VH). Variation due to area type is represented by including three values (one for each type of area), to indicate the magnitude of the effect a performance parameter has on each of the six elements of strategy.

The coarse quantization in table (6.3) is brought about by the qualitative nature of the constraints, as evident from the examples given above. It is also in accordance with the symbolic type of processing pursued by this system, as evident in the knowledge and control rules. The fuzzy nature of the constraints, and of their combined influence in specifying the elements of local strategy, is once again evident. Each constraint is best represented by a direction of influence, and a symbolic qualifier that resembles the linguistic hedges discussed in section 6.4.1.

The concepts of fuzzy logic will now also be extended to the problem of local strategy determination. Unlike the global strategy, more than one decision is required. Six elements must be instantiated in order to completely specify the local strategy. Therefore, as many decisions are needed to assign a state to each of these elements. The model F is thus composed of six independent, yet identical, decision processes, that can be executed in parallel. Each process involves the selection of one of a finite number of states, and hence the determination of one component of the local strategy vector.

The strategy vector for area α can be written as

$$\bar{S}_{\alpha} = [S_1, S_2, S_3, S_4, S_5, S_6] \quad (6.48)$$

The following analysis will apply equally to each of the strategy elements.

Let the set of n states for strategy element S_i be given by

$$S_i = [s_{i1}, s_{i2}, \dots, s_{in}] \quad (6.49)$$

We can construct M fuzzy subsets of S_i . Each subset corresponds to one of $M = 2m+3$ parameters constituting an area's performance vector. Each subset will have the form

$$Y_k = \left[\frac{\mu_{k1}}{s_{i1}}, \frac{\mu_{k2}}{s_{i2}}, \dots, \frac{\mu_{kn}}{s_{in}} \right], \quad 1 \leq k \leq M, \quad (6.50)$$

where μ_{kj} is the degree by which performance parameter k satisfies the requirements of state j for strategy element i in the given area. Each parameter is evaluated as a numerical quantity in the interval $[0,1]$. If a parameter has a positive effect on a state, then the higher the value of that parameter, the more the degree of satisfaction of that state with it. For a negative effect, that degree would increase with lower parameter values. If parameter k is found to have a value p_k , then

$$\mu_{kj} = \begin{cases} p_k & \text{if } k \text{ has a positive effect on } j, \\ (1 - p_k) & \text{if } k \text{ has a negative effect on } j, \end{cases} \quad (6.51)$$

as specified by the entries of table (6.2).

The decision objectives represented by the M fuzzy subsets of equation (6.50) can now be computed from the performance parameters of the area. They can then be used to compute a fuzzy decision set D_i on the states of strategy element i , so that

$$D_i = \bigcap_{k=1}^M \gamma_{ik} \quad (6.52)$$

where γ_{ik} is a quantity that reflects the relative importance of performance parameter k , in determining strategy element i , according to table (6.3). Recalling the discussion in section 6.4.1. about hedges, a positive power that is greater than unity, will emphasize the influence of a particular objective. A fractional power will have the opposite effect. Zadeh uses the power of 2 to represent the hedge "very" [Zadeh, 1972]. In table (6.4), we extend this to account for all the symbolic entries of table (6.3). The values of γ_{ik} will thus be equal to the numerical entry in table (6.4), that corresponds to the entry for parameter k and element i , for the given type of area in table (6.3).

Symbol	Meaning	Value
VL	very low	0.5
L	low	0.75
M	medium	1.0
H	high	1.5
VH	very high	2.0

Table (6.4) : Symbol table for hedges.

Table (6.3) indicates that the relative importance of the performance parameters varies from one strategy element to the next. It also depends on the type of area for which the strategy is determined. The effect of area type is thus indirectly incorporated into the decision-making process, by redefining the emphasis on different performance parameters.

The first four strategy elements in table (6.2) are binary valued. The entries that are positively supporting one state, are seen to have a negative effect on the other state for these elements. The last two elements are multi-valued, and each parameter can support more than one state, in a positive or negative manner, simultaneously. It is the integrated effect over all parameters that tips the balance towards a particular state. This is, of course, a mutually exclusive process, since a strategy element can only assume a single state at a time. The fuzzy decision set D_i computed in equation (8.52), provides the degree of satisfaction of the decision-maker with each of the states of strategy element i . The state with the maximum satisfaction value is chosen to represent that element in the output strategy vector. This process is repeated for all six strategy elements.

The SUPERVISOR employs the strategy rules to execute dynamic strategy selection. A strategy rule is constructed for each state of every strategy element. The conditions correspond to the fuzzy constraints on the performance parameters. The action assigns the state associated with the

rule, to the strategy element. All the rules are evaluated. The condition that is least satisfied for each rule represents the degree by which that rule matches the performance measures. The rule with the best matching assigns its state to the strategy element. Appendix C describes how the strategy rules are constructed from the entries in tables (6.2) and (6.3).

The model F in figure (6.3) can thus be represented by a set of modules which compute the degree of satisfaction of each strategy element state, with the value of each performance parameter. These are followed by a maxi-min selection module for every element, as shown in figure (6.4)

6.5. CONCLUSIONS

In the previous sections, we have discussed the various factors involved in setting the strategy of our rule-based image segmentation system. We have demonstrated the difficulties involved in getting such a complex yet versatile system to function in an optimized fashion. To do this, we have analyzed the system strategy into basic, well defined, and independent elements, that can acquire a number of finite and specific states. Next we defined a set of parameters that measure the performance of the system, as reflected by the state of the output at any point during processing. A model was then introduced that used these parameters in evaluating decision functions to allocate a specific state to each strategy element.

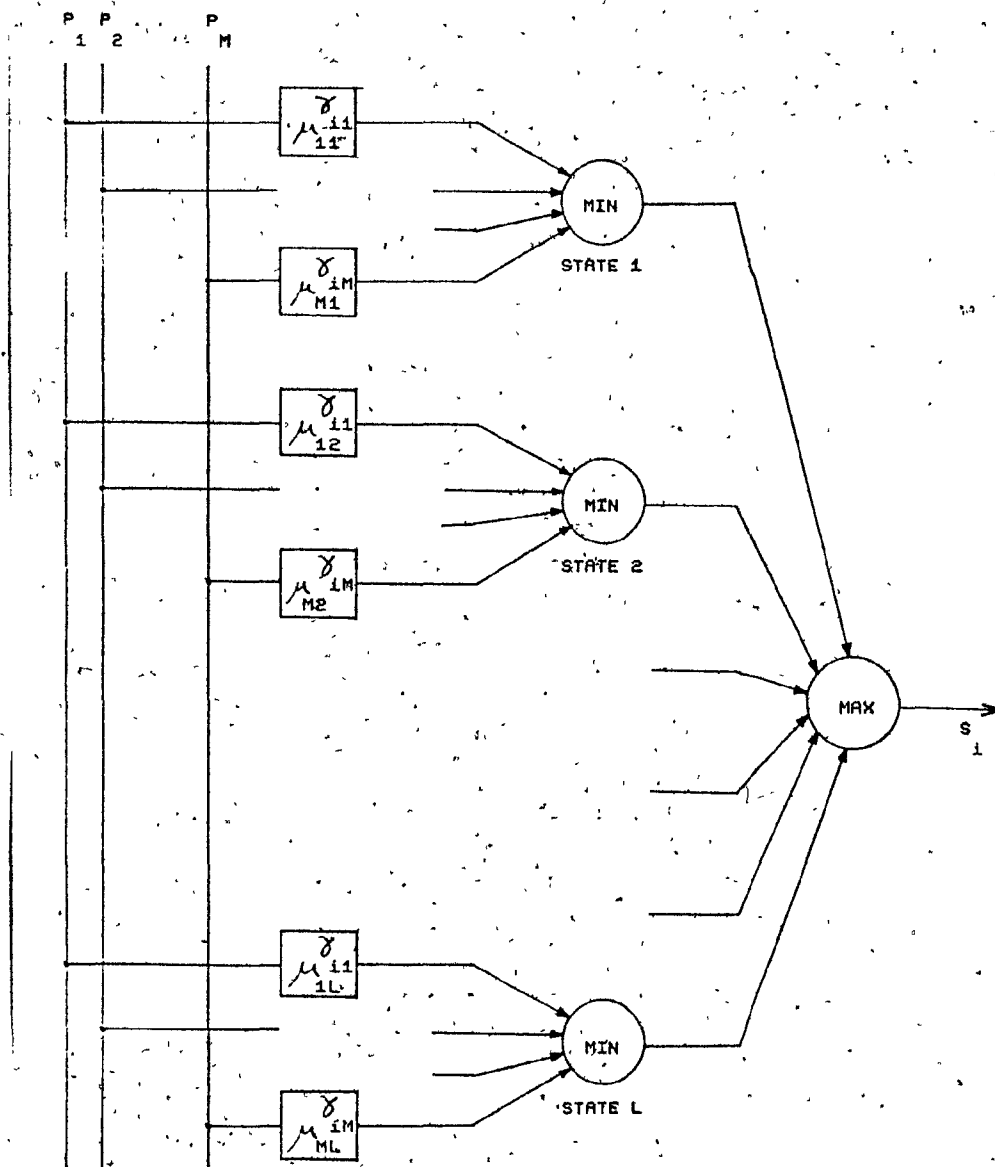


Figure (6.4) : The solution to model F.

The solution proposed involves a novel approach to both conflict resolution in a rule-based production system, and data selection in image processing systems. The system is designed to allow for dynamic setting of its rule matching priorities and its focus of attention scheme. A pool of

focus of attention areas is generated according to certain criteria embedded in rules that are designed for this purpose. The performance parameters discussed in section 6.3 are subsequently computed for each area, based on the initial segmentation status within that area. These parameters are stored in the STM, and are dynamically updated during processing, so that they can continuously and appropriately represent each area, at any point in time. The FOCUS OF ATTENTION MODULE selects the most appropriate area to work on at any time. The criteria used are those of processing need and uniform time sharing. Once an area is selected, the SUPERVISOR proceeds to define the strategy within that area. It evaluates decision functions that reflect the effect of the performance parameters of the area, on each of the individual strategy elements. Both area selection and strategy evaluation employ a fuzzy decision-making methodology. This accounts for the inherent imprecision in the constraints representing the relation between the data and the resulting decisions.

One aspect of the system strategy was shown to be related to the problem of data selection for a large complex system. This problem is significant, as it affects both the quality of the result and the speed of reaching that result. The solution provided here is both dynamic and data-driven. The selection of the next area of attention, as well as the next region and line in that area, is suggested by the state of the segmentation as reflected by the data. Their order

will change when the data changes. The underlying criteria are those of efficiency and minimum error. This is reflected in preferring image areas that will lead to a larger improvement in the state of the segmentation.

Another aspect of strategy setting defined rule priorities for conflict resolution in the rule-based image segmentation system. Through a similar hypothesis, a dynamic data-driven rule ordering was presented. The order in which the rules are matched by the system depends on the state of the data. It is continuously updated to emphasize the relative importance of various knowledge and control criteria, according to the effect that they will have on the data. In other words, rules that are more useful are given more preference. The term "useful" here is measured in terms of the influence these rules will have on the state of segmentation as reflected by the data.

The system is designed in an extremely modular and flexible manner, that exhibits both knowledge and control separability. This enables it to dynamically alter each of its strategy elements in an easy and straightforward fashion. By implementing this dynamic strategy in our rule-based system, we are serving a dual purpose. First, we ensure efficient processing by modifying the areas in the image in an appropriate order. The second goal is to minimize the segmentation errors by selecting the best rules at the proper point in the computation. The next chapter discusses the system implementation and the experimental results.

CHAPTER 7

RESULTS AND CONCLUSIONS

7.1. SYSTEM IMPLEMENTATION

The rule-based image segmentation system described in the previous chapters has been implemented on the VAX 11/780 of the Computer Vision and Graphics Laboratory at McGill University. The machine has 3 Megabytes of physical memory and operates with a virtual memory system. Pictures were scanned off transparencies (typically 35 mm film) using an Optronics drum scanner with a maximum resolution of 50 microns, and a maximum sensitivity of 256 grey levels. Images used in the following experiments were stored in 256 by 256 arrays with 6 bits per pixels. The output was viewed on a TV monitor interfaced to a Grinnell color graphics memory system with 256 by 240 resolution, 3 color channels, and a 64 grey level per channel sensitivity.

A collection of routines named MARBLES (for McGill Automatic Rule-Based Low Level Segmentation) constituted the backbone of the system. They performed a variety of functions including the image input, feature computation, rule matching, strategy setting and data modification. Figure (7.1) is a block diagram of the system implementation. In addition to automatic processing, user interaction was allowed for experimental purposes. The model designer could access and modify the rule-based model stored in the LTM. The image and segmentation data could also be examined in order to allow a better understanding of when and why different rules became effective. Moreover, the data could be graphically displayed so that a visual evaluation of region and line partitions was possible.

Experiments with the system were designed to fulfil two main purposes. The first was to test and evaluate the general purpose knowledge that is used to partition an image. The performance of the system was measured when different combinations of knowledge rules were employed. Some rules were modified to achieve better results, while others were discarded if they proved to be useless. The second purpose was to assess different control strategies under varying data conditions. The appropriate subset of control rules that simulated a particular processing strategy were selected and tested. The resulting performance measures were recorded and compared in order to determine the best strategy for each data situation. The results were used to set up the decision

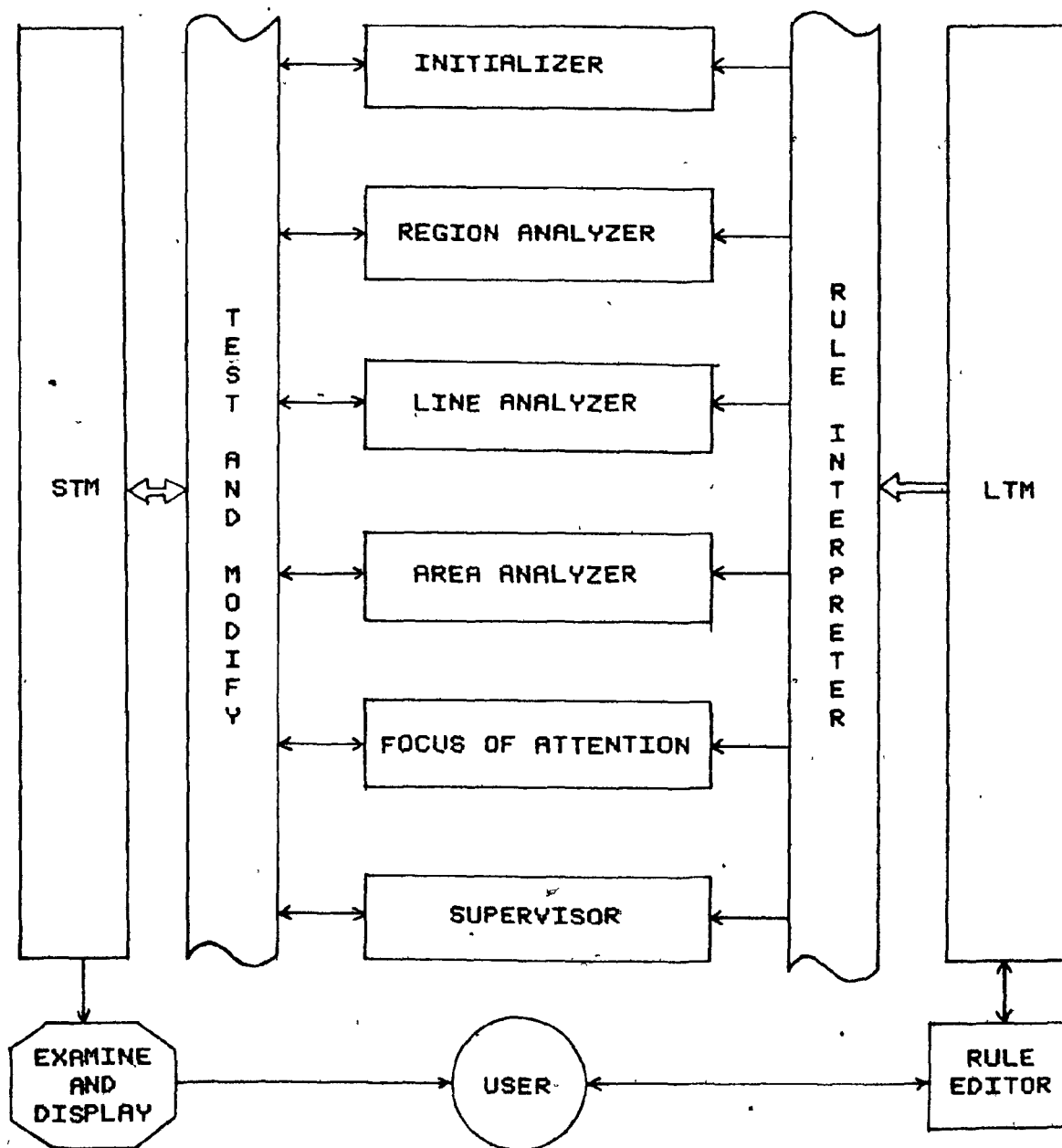


Figure (7.1) : Block diagram of system implementation.

criteria for dynamic strategy evaluation presented in tables (6.2) and (6.3).

The next two sections describe the experiments undertaken for knowledge and strategy evaluation. This is followed by an account of the results obtained for different focus of attention areas. Finally, the system performance with an optimal knowledge base and a dynamic strategy is compared to other segmentation techniques in the literature. The thesis concludes with suggestions for future research and recommended additions that are based on the gained experience with this system.

7.2. KNOWLEDGE BASE EXPERIMENTS

In chapter 5, a set of region and line analysis rules was presented that is based on previous efforts and general Gestalt rules of grouping. The segmentation error measures described in chapter 3 and the performance measures introduced in chapter 6 were used to evaluate the validity of these processing criteria. When the program was executed with one set of rules the error and performance measures changed with each action taken. The following classes of measures were recorded each time a rule fired:

(1) The under-merging, over-merging, and combined distance error measures with respect to a hand segmented region map according to equations (3.26), (3.28), and (3.29), respectively.

(2) The missing lines, additional lines, and combined line error measures according to equations (3.31), (3.32), and (3.29), respectively.

(3) The region uniformity and region contrast performance measures according to equations (6.7) and (6.12), respectively.

(4) The line contrast and line connectivity performance measures according to equations (6.18) and (6.19), respectively.

(5) The number of matching cycles up to the current action.

The first four classes contributed to ten possible curves when each measure was plotted as a function of the number of actions. These provided profiles of the system performance as the processing progressed. In order to maintain a concise presentation, only curves that exhibited interesting behaviour will be given for each experiment. The fifth class is directly related to how the processing terminated. When the number of actions was plotted against the number of cycles (including those where none of the rules fired), the result was a profile of the activity of the particular rule set used. This curve had the typical form shown in figure (7.2). Near the beginning of processing, a rule fired in almost every cycle. However, as the data approached its final state, the number of rule matches declined rapidly and the curve saturated when no more actions were possible. In all of the following experiments processing ended when saturation was detected within the actions-versus-cycles curve. In terms of the results that follow, the average times for matching conditions and executing actions on the VAX 11/780 were 0.4 and 1.3 seconds, respectively.

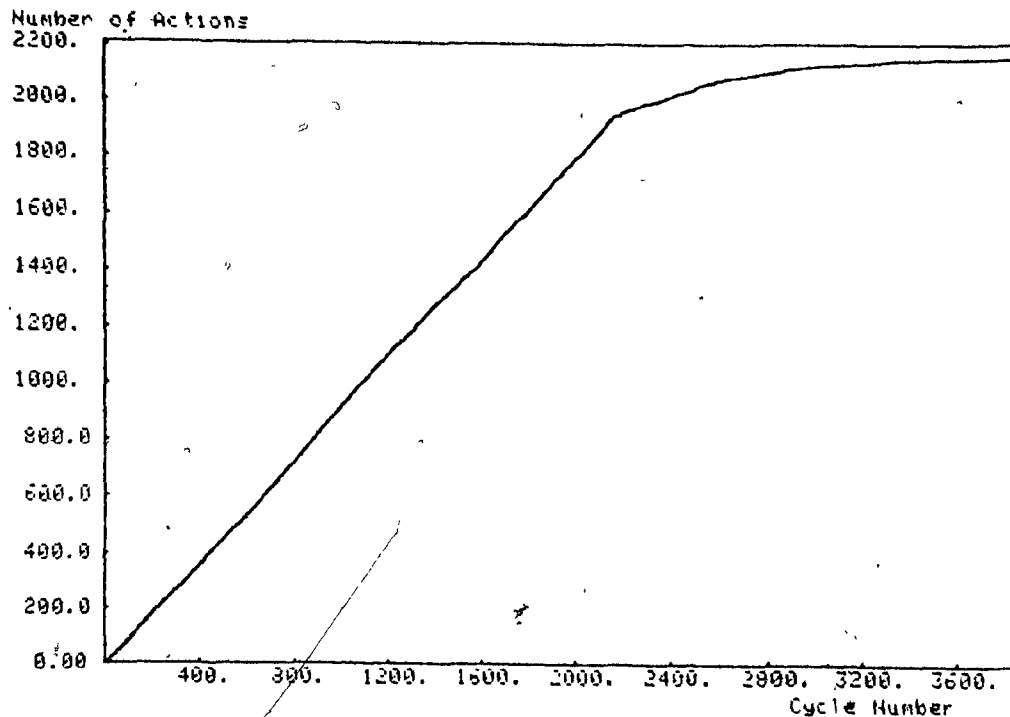


Figure (7.2): Actions-versus-cycles curve.

A combined final measure of performance and activity was obtained by dividing the change in total error during one experiment by the total number of rule cycles. This measure was used to compare a set of experiments when both the numerator and denominator were normalized by their maximum values within the set. The value F for each experiment within a given set was computed as follows:

$$F = \frac{M_{\text{initial}} - M_{\text{final}}}{M_{\text{max}} - M_{\text{min}}} \times \frac{\text{Max. no. of cycles}}{\text{No. of cycles}} \quad (7.1)$$

where the maximum and minimum values of the error measure were calculated over the given set of experiments.

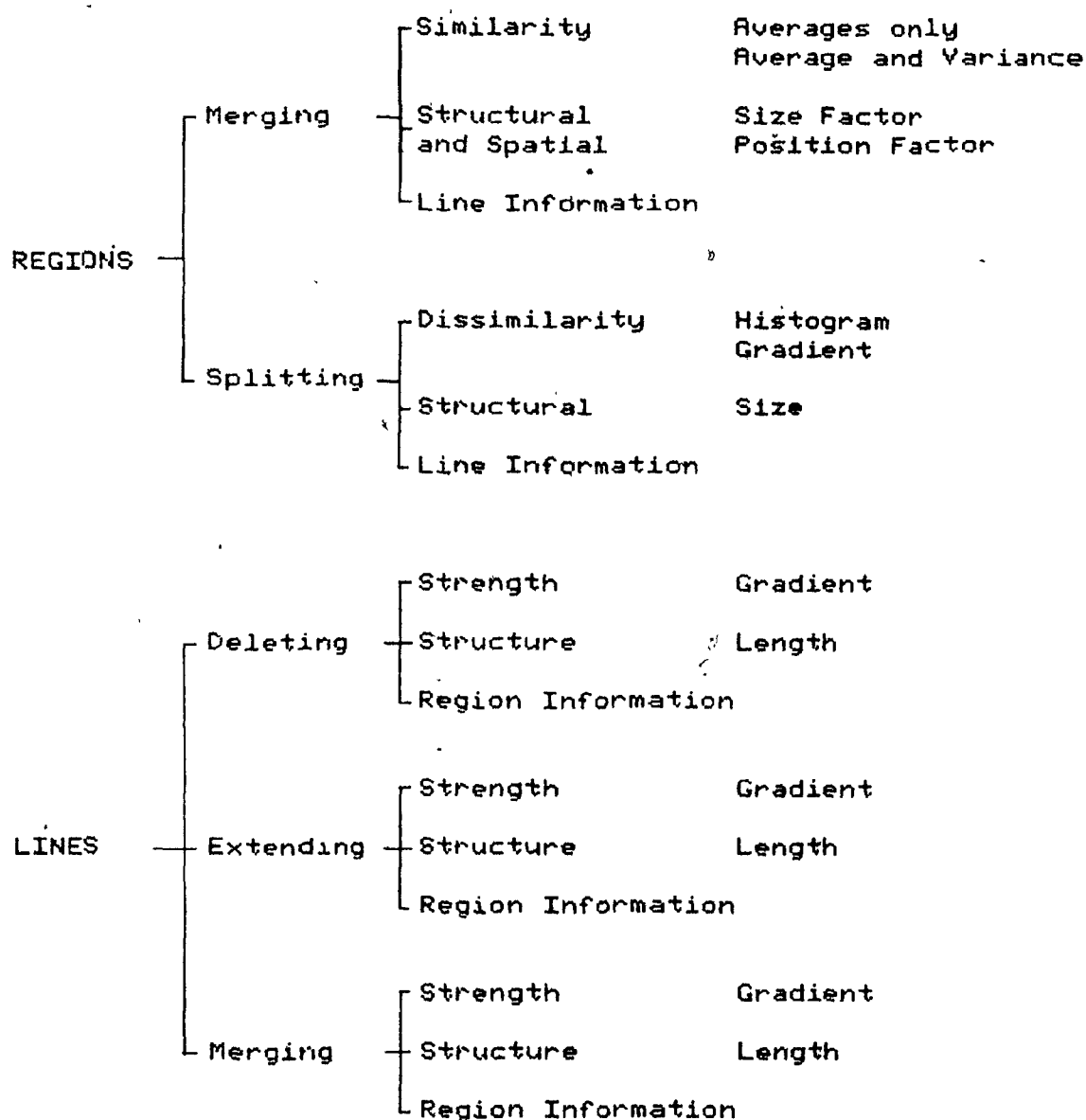


Figure (7.3) : Rule categories.

The knowledge rules were divided into the categories shown in figure (7.3) according to the basic criteria that they represent. Different experiment sets compared and ranked the rules within each category. Each set contained experiments that started with the same initial partitions of regions and lines, used the same control strategy, but

different combinations of knowledge rules. In addition to the performance curves described above, the appropriate output partitions (regions or lines) were compared, and a table of initial and final performance values for each experiment set was also provided.

A window in the image of the electric motor shown in figure (7.4) was used for the experiments. The window, together with the hand segmented region partition, are shown in figure (7.5). Following is a detailed discussion of the results. For readers who wish to skip the details, section 7.2.8 provides an overall ranking of the knowledge rules based on the experiments described in sections 7.2.1 to 7.2.7.

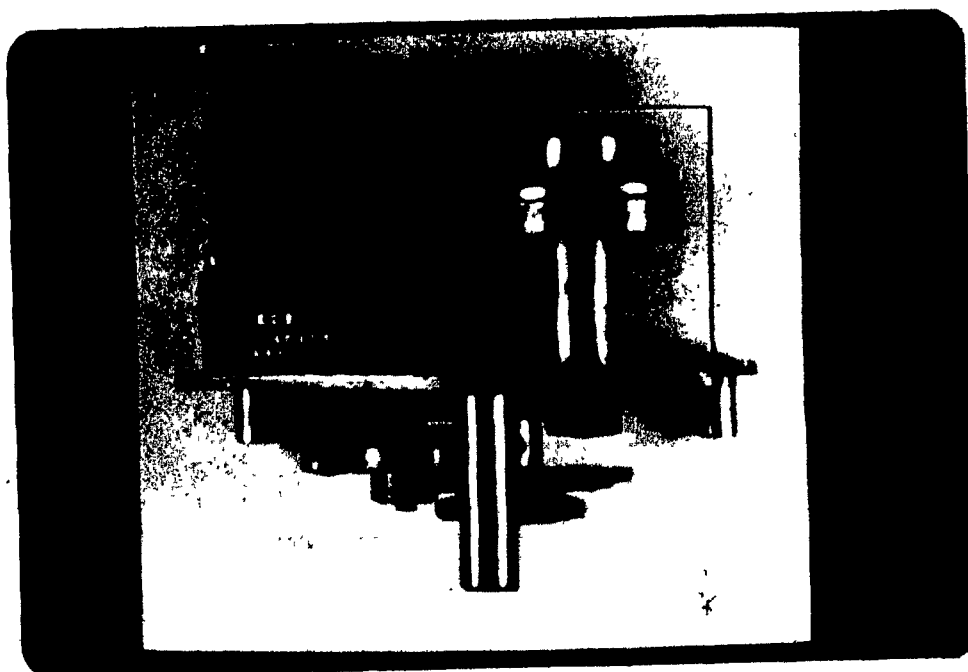


Figure (7.4) : A window in the image of a motor.

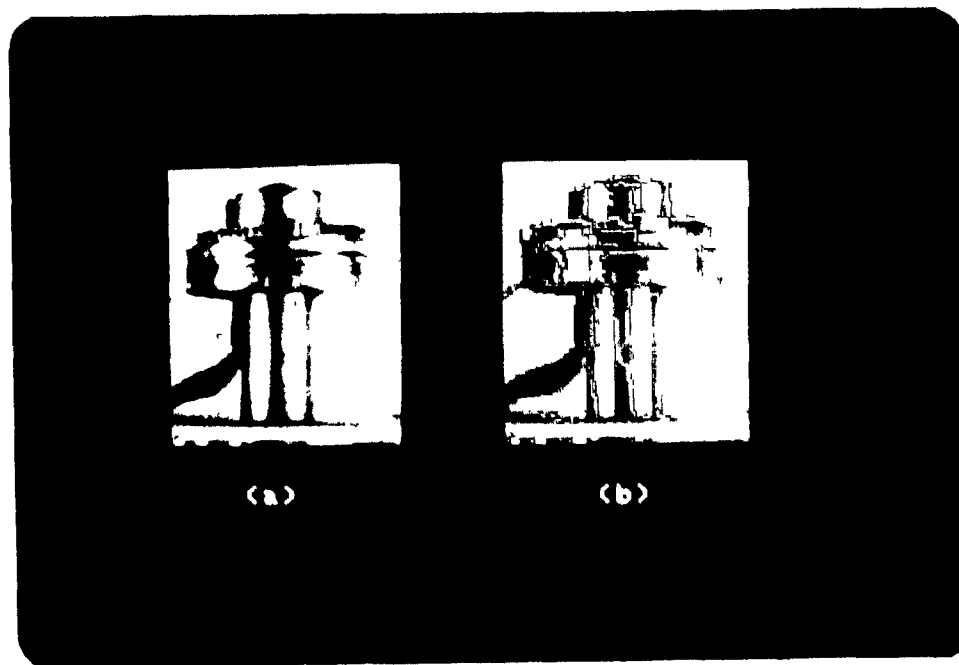


Figure (7.5) The selected window
 (a) The original image
 (b) A hand segmented region map

7.2.1. Region Merging Experiments

Starting with 2×2 windows as initial regions, the first experiment employed rule (801), which merges adjacent regions based on the similarity of their average features. Experiment 2 allowed only the regions with similar average and variance of features to merge, according to rule (808). The latter is thus a more conservative approach to region grouping. Experiment 3 relaxed the criteria for merging regions of small size, by adding rules (802), (806), and (807) to rule (808). Therefore, a cleanup of small regions was added to the similarity grouping process. Experiment 4

included rules (803), (804), and (805) together with rule (808) in order to eliminate the small regions that fall at the boundaries between larger ones

Figures (7 6a) to (7 6d) show the resulting region partitions for the four experiments, and table (7 1) summarizes their performance. The combined error measure, shown in figure (7 7a), declined steadily at a relatively slow rate when small regions were initially merged together. Changes became faster near the end when the sizes of the regions became much larger so that each merging action had a more global effect on the measure. Figures (7 7b) and (7 7c) represent the curves of the under-merging and over-merging components of the measure, respectively. Note that the decline in the under-merging error was accompanied by an increase in the over-merging error, yet the net result was a decline in the combined error measure.

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
	Initial	0	0.95	0.26	92.41	3.60	65.46	0.00
1	2800	2370	0.85	0.95	9.56	41.70	30.25	1.46
2	3865	2155	0.89	0.78	36.95	28.70	33.08	0.97
3	3027	2315	0.85	0.85	39.72	32.30	36.20	1.12
4	4106	2169	0.89	0.86	36.24	30.90	33.68	0.90

Table (7.1): Region merging experiments.

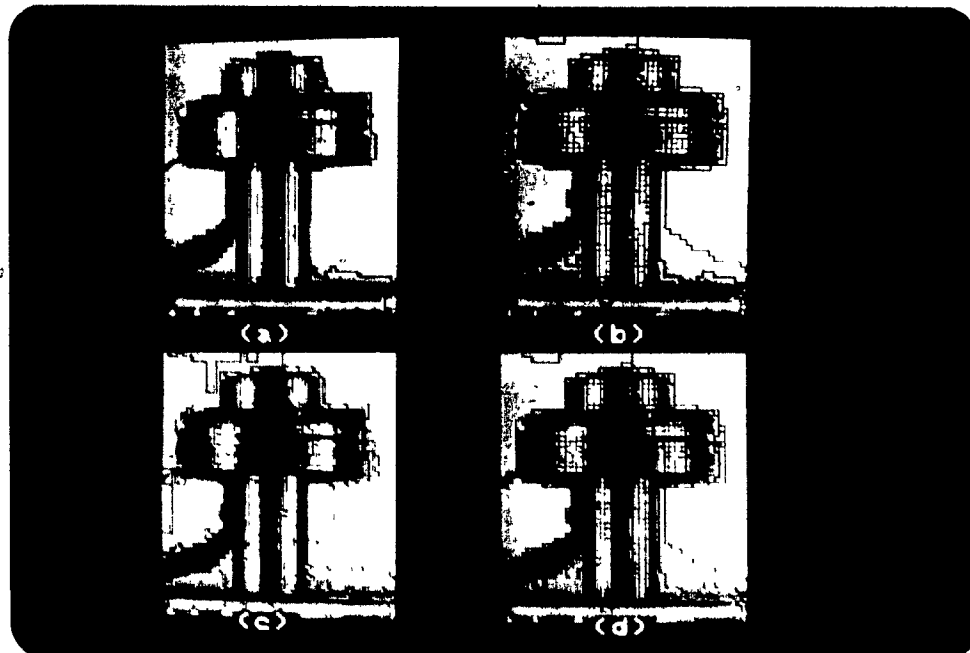


Figure (7.6) . Region maps for merging rules experiments
 (a) Exp 1 Similarity of average features
 (b) Exp 2 Similarity of averages and variances
 (c) Exp 3 With cleanup of small regions
 (d) Exp. 4 With cleanup of boundary regions.

The curves for experiment 1, reflect its less conservative nature, so that the decline in the under-merging error is more pronounced, but at the cost of a larger increase in the over-merging error. In all, rule (801) performed better than rule (800), with the largest decline in the total measure, and the lowest number of cycles. The cleanup rules of experiment 3 produced an adverse effect on the output. They resulted in a larger overall error, although they improved the ratio of the number of actions to the total cycles, and thus yielded a higher value for F .

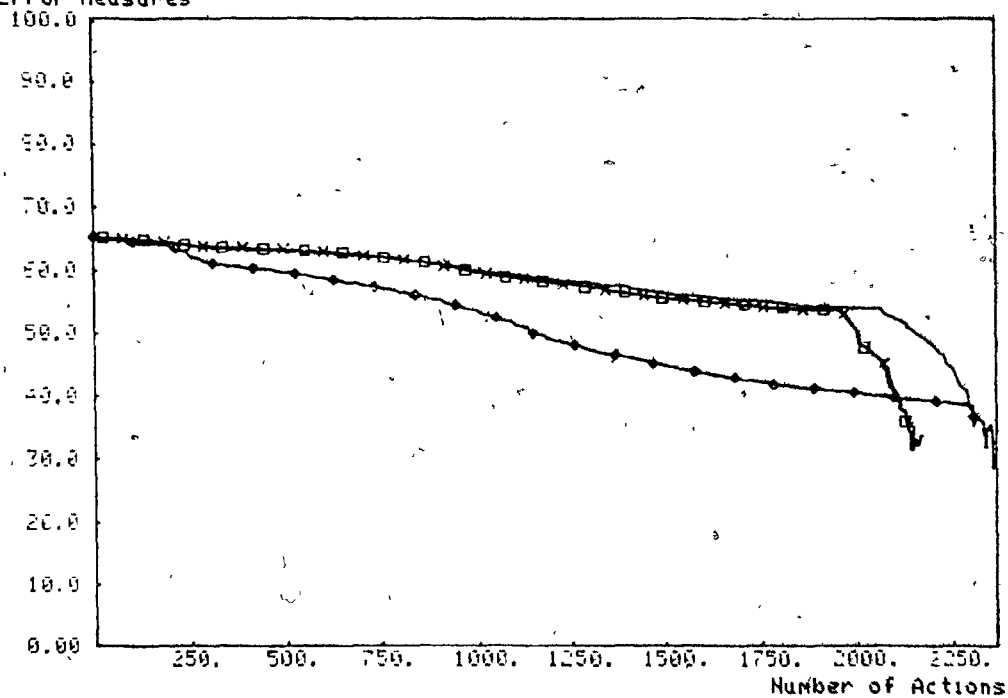
Although these cleanup criteria removed unwanted small regions, their relaxed constraints caused other errors in the final partition.

The additional rules in experiment 4 had almost no effect on performance. This is apparent from the fact that the curves of experiments 2 and 4 almost coincided. The only influence these rules had was to increase the number of matching cycles with no similar increase in activity, thus reducing F.

The region uniformity and contrast curves are shown in figures (7.7d) and (7.7e), respectively. Uniformity declined steadily with merging, and more so for experiment 1 than for 2. The curves for 2 and 4 remained very close, and the curve for 3 was worse than that for 2. On the other hand, contrast for experiment 1 showed a higher improvement than that for 2. Note, however, that the curve for 3 still did not do any better.

An important observation here is that the performance measures of figures (7.7d) and (7.7e) behaved in a consistent manner with the segmentation error measures of figures (7.7a) to (7.7c). This applies to all the experiments conducted with the system, and will also hold true for the line dependent measures whose behavior will be demonstrated in experiments that follow. Such consistency can only attest to the validity of using these performance measures as feedback criteria that dynamically adjust the processing strategy.

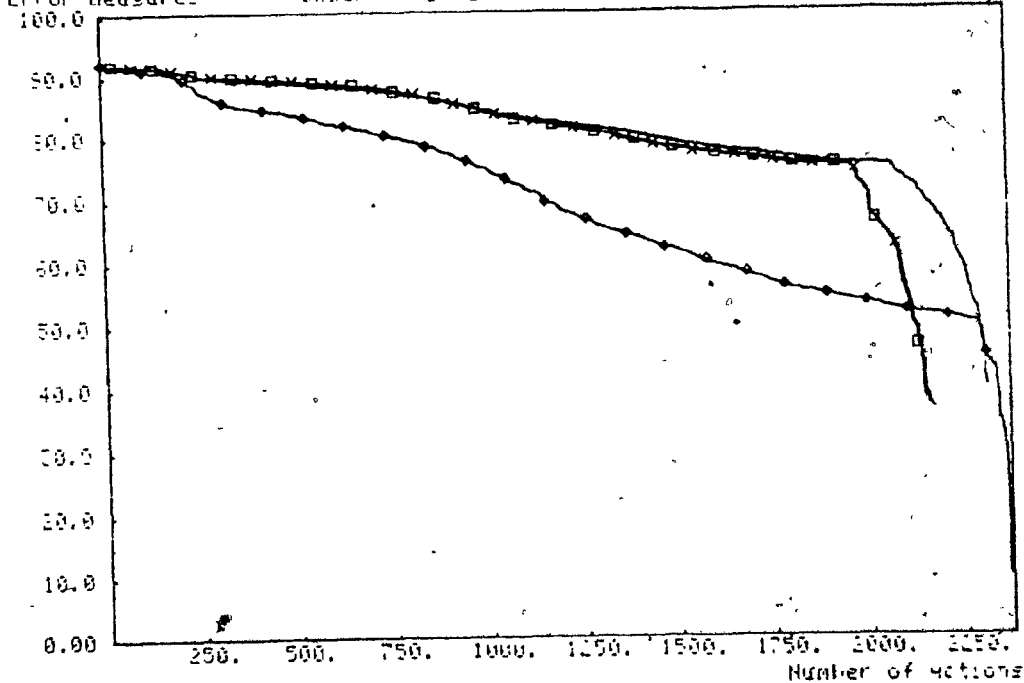
Error Measures



(a)

Error Measures

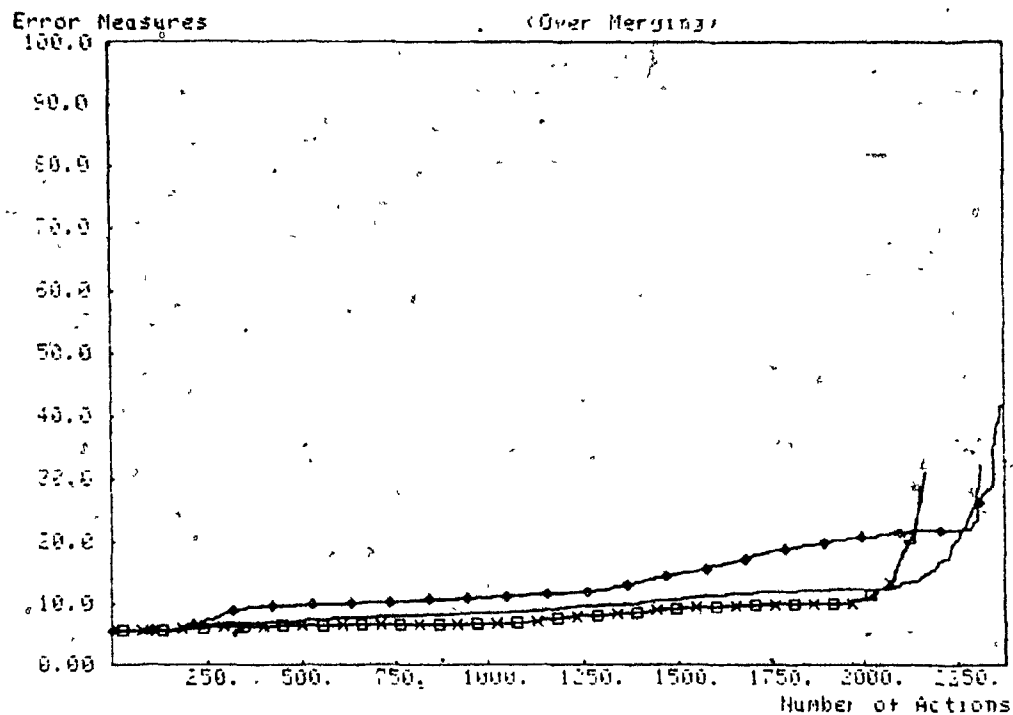
(Under Merging)



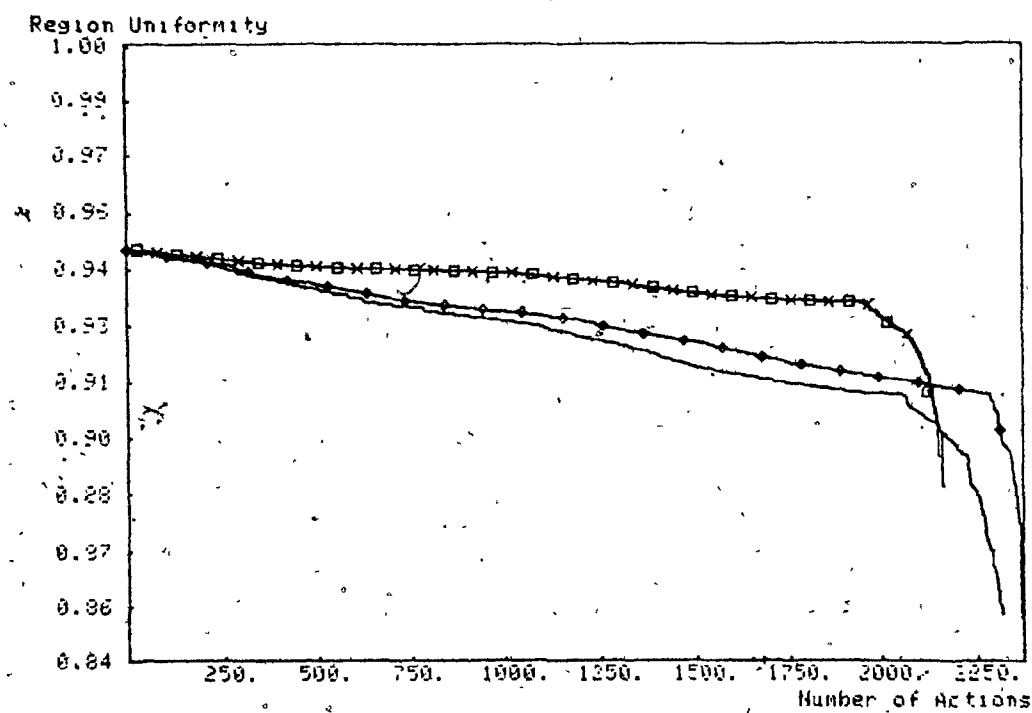
(b)

Figure (7.7): Merging experiments performance measures.

- Exp. 1 $\diamond$
- Exp. 2 $\square$
- Exp. 3 $-$
- Exp. 4 $\times$



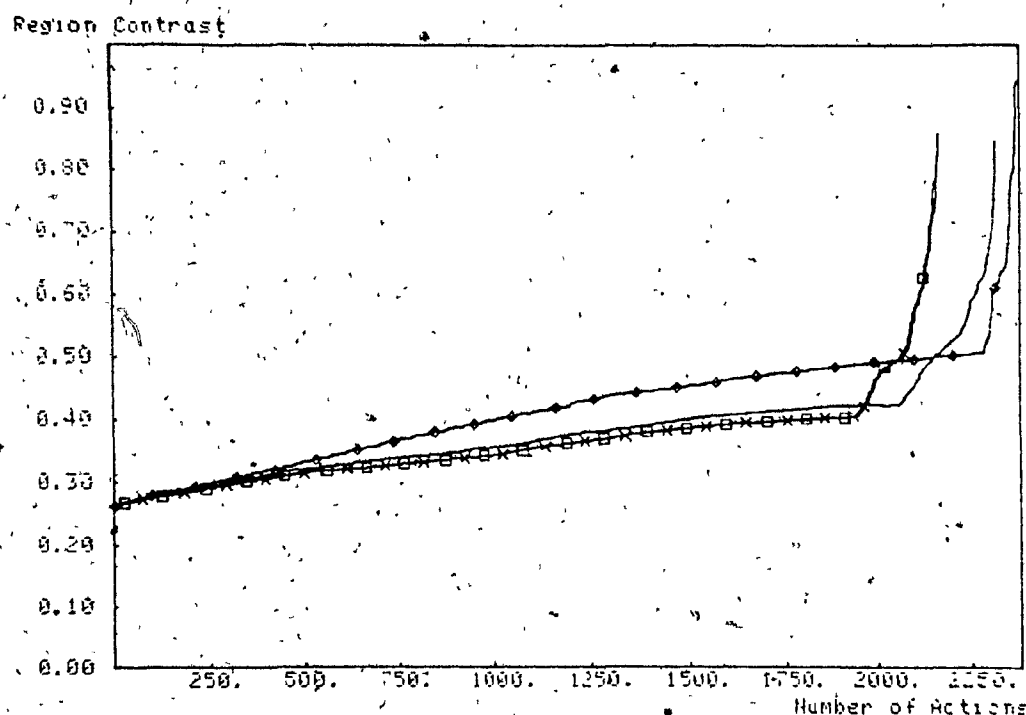
(c)



(d)

Figure (7.7) contd. :

Exp. 1	◇
Exp. 2	□
Exp. 3	—
Exp. 4	x



(e)

Figure (7.7) contd. : Exp. 1 $\diamond$
 Exp. 2 $\square$
 Exp. 3 —
 Exp. 4 $\times$

7.2.2. Region Splitting Experiments

The same image was used to test rules that split regions, except that now the whole window was initialized into one region. Experiment 5 employed rule (901) which split a region if its histogram is bimodal. Experiment 6 used the more restrictive rule (902), which imposes a lower limit on the degree of bimodality of the histogram for a split to occur. The above rule was substituted by two rules (903) and (904) in experiment 7 that relate region size to the degree of histogram bimodality. Finally, in

experiment 8, splitting was subject to the additional restriction of a high average gradient value over the region. The two rules that accomplished this for different size regions are (906) and (907).

The resulting region partitions for experiments 5 to 8 are shown in figures (7.8a) to (7.8d), respectively. Table (7.2) summarizes performance with respect to the initial values. It is seen that splitting that is based on the histogram alone provided the worst output. In fact, the decrease in over-merging error was more than compensated for by the increase in the under-merging error, with a net increase in the total error. This is also evident in the unnecessary partitions in the image of figure(7.8a). Restricting the degree of bimodality helped the situation by decreasing the amount of under-merging, but still caused more harm than benefit. Entering the size as a factor caused further improvement to the resulting under-merging error, but still fell short of delivering a plausible output. On the other hand, experiment 8 produced a considerably better output, mostly due to the fact that the high gradient restriction limited the under-merging error to a much lower value than was possible using the other rules.

The under-merging and over-merging error curves are shown in figures (7.9a) and (7.9b), respectively. The superior performance of rules (906) and (907) is reflected in the achievement of the same level of improvement in the over-merging error in a considerably smaller number of

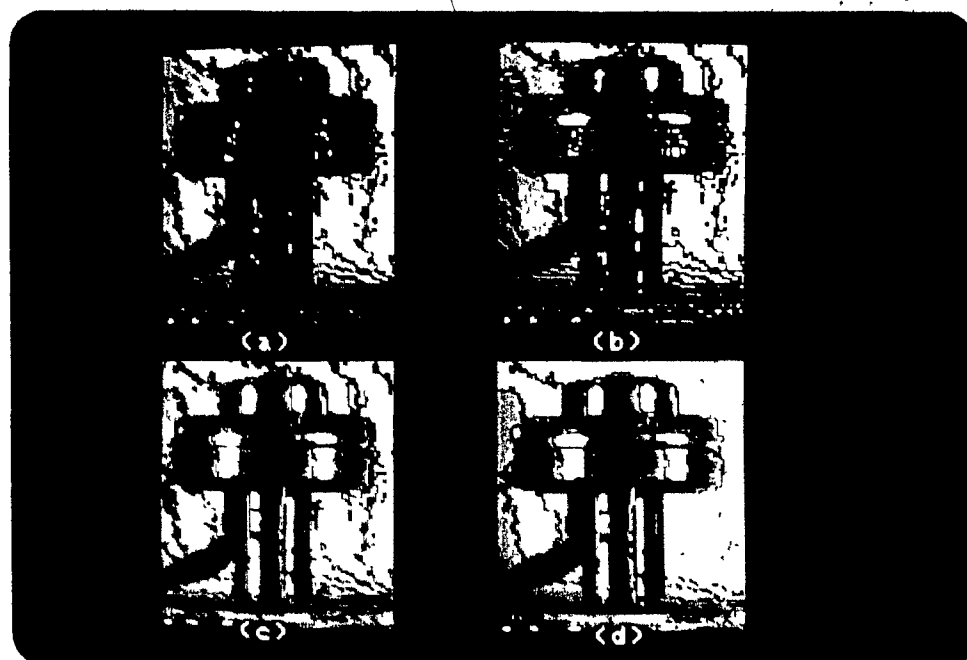


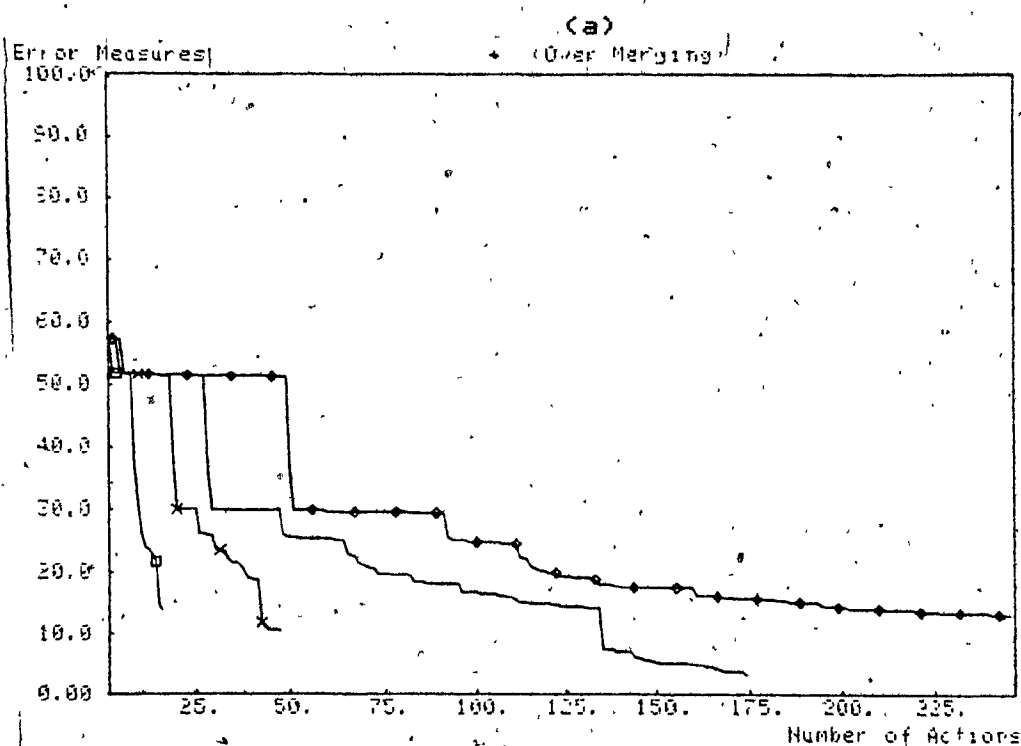
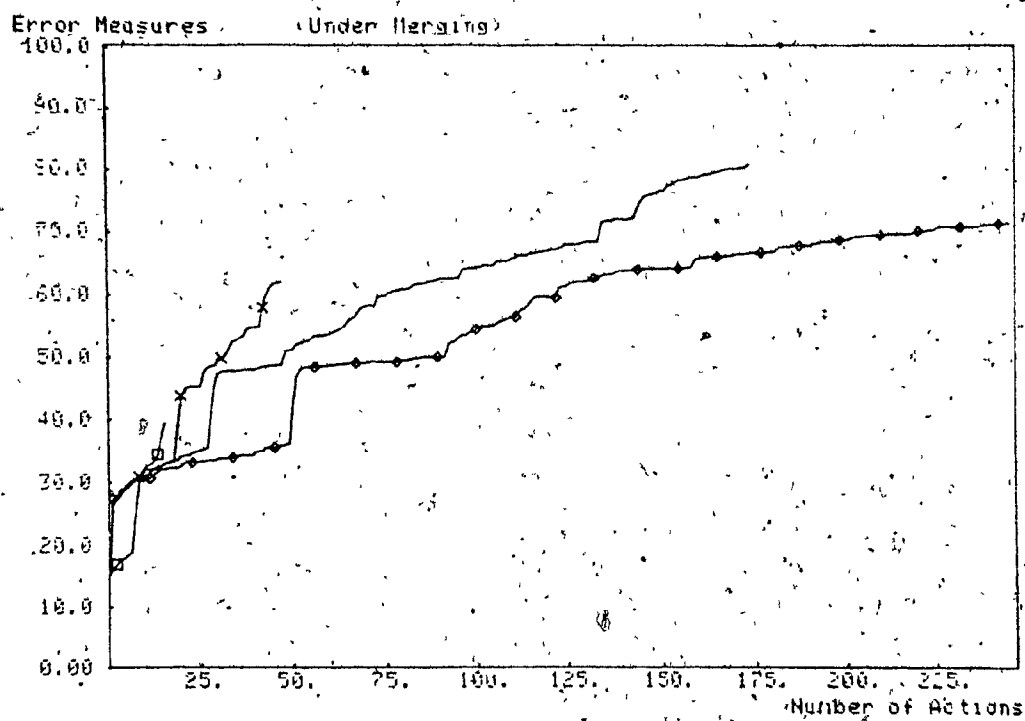
Figure (7.8): Region maps for splitting rules experiments.

- (a) Exp. 5 : Splitting if histogram is bimodal.
- (b) Exp. 6 : Lower limit on bimodality.
- (c) Exp. 7 : Using bimodality and size.
- (d) Exp. 8 : Using gradient, bimodality, and size.

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
	Initial	0	0.67	0.54	0.00	59.30	41.93	0.00
5	1116	174	0.97	0.35	81.01	3.30	57.33	-0.55
6	1001	246	0.98	0.38	71.35	13.10	51.30	-0.37
7	337	47	0.94	0.39	62.16	10.40	44.56	-0.31
8	133	16	0.92	0.47	39.32	14.00	29.51	3.74

Table (7.2): Region splitting experiments.

actions, and with a much lower increase in the under-merging error. This is also reflected in the increasing value of parameter F for experiments 5 through 8.



(b)

Figure (7.9): Splitting experiments performance measures.

Exp. 5 ◇
 Exp. 6 ▽
 Exp. 7 ×
 Exp. 8 □

7.2.3. Initial Region Size Experiments

By examining the results of the experiments presented in the previous two sections it is easy to theorize that a combination of merging and splitting rules will improve performance over the use of each set individually. Splitting rules alone tend to under-merge the image, especially if they are based on the histogram. Merging rules can either be too conservative, or will cause over-merging errors, while in both cases requiring a large number of matching cycles, because of the large number of initial regions.

In an attempt to verify the above theory, the best merging rule (801) and the best splitting rules (906) and (907) were selected. A question arised as to the size and number of regions at the beginning of processing. Because both splitting and merging were now employed, different initial configurations were possible. To test the effect these had on system performance, grids of square regions were generated in the form shown in figure (7.10a). Seven experiments were performed using the above three rules on initial partitions of squares with various sizes. They ranged from a 2X2 grid to a single region image, and their sizes and numbers are shown in table (7.3).

The resulting partitions are shown in figures (7.10b) to (7.10h), and the performance measurements are summarized in table (7.4). The quality of the output partitions deteriorated with the increase in the size of the initial

squares. This was followed by a remarkable recovery when the size reached a maximum that covered the whole window. This is also evident in the graphs for the total error measure shown in figure (7.11). The curves for experiments 9 and 15 terminated at lower error values than the curves for experiments with intermediate initial sizes.

Experiment	Size	Number of regions
9	2 X 2	2438
10	4 X 4	621
11	8 X 8	168
12	16 X 16	42
13	32 X 32	12
14	54 X 54	4
15	90 X 106	1

Table (7.3) : Initial region sizes and numbers.

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
9	3344	2388	0.84	0.98	21.81	26.30	24.16	0.77
10	1596	614	0.81	0.96	39.54	30.10	35.14	1.38
11	617	156	0.70	0.75	31.55	31.30	31.43	3.78
12	533	165	0.80	0.98	34.67	34.60	34.64	4.17
13	444	142	0.86	0.93	36.11	32.00	34.12	5.05
14	440	125	0.89	0.81	37.46	31.80	34.75	5.04
15	468	111	0.89	0.80	26.34	25.00	25.68	5.40

Table (7.4): Region Size Experiments.

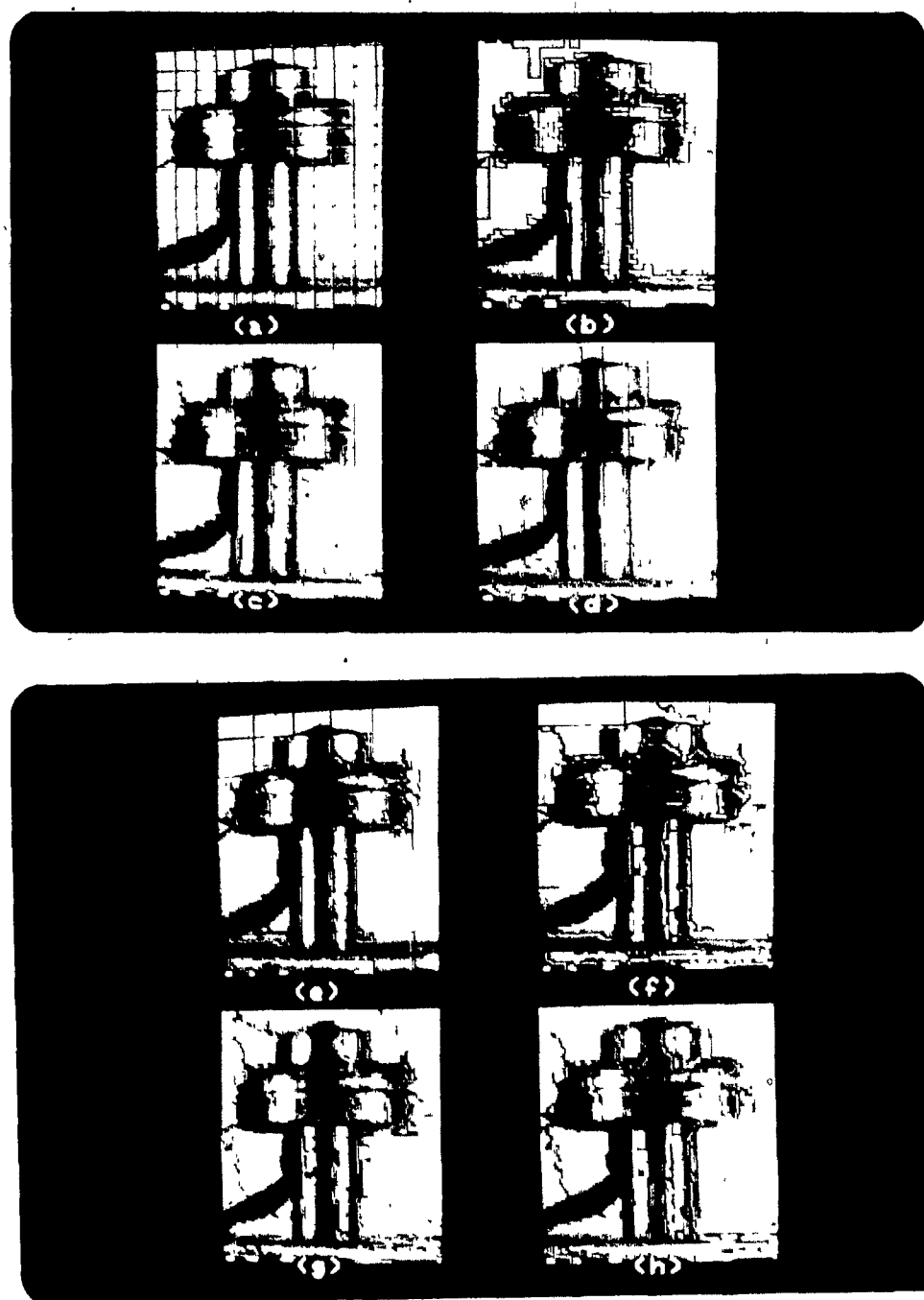


Figure (7.10): Region maps for initial size experiments.

- (a) Example of a square grid of initial regions
- (b) Exp. 9 2438 initial regions of size 2X2
- (c) Exp. 10 621 initial regions of size 4X4
- (d) Exp. 11 168 initial regions of size 8X8
- (e) Exp. 12 42 initial regions of size 16X16
- (f) Exp. 13 12 initial regions of size 32X32
- (g) Exp. 14 4 initial regions of size 54X54
- (h) Exp. 15 1 initial region of size 98X106

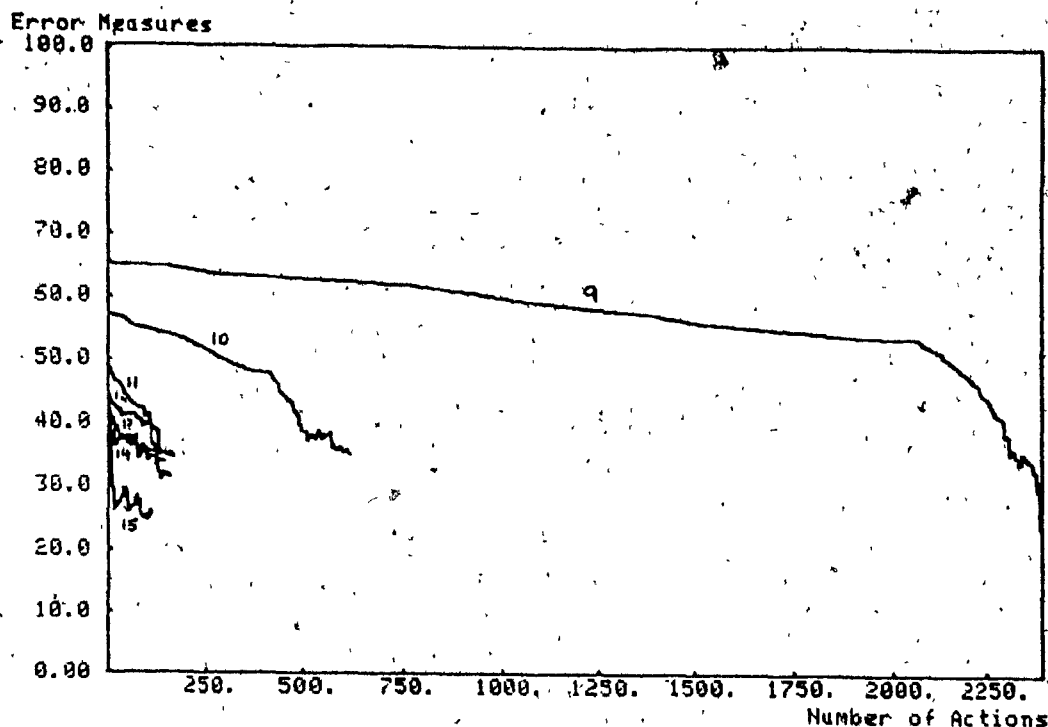


Figure (7.11): Total error measures for region size exps. The curves shown are for experiments 9 to 15. Note that the curves for 9 and 15 end at lower error values than the rest.

These results are explained by the fact that the presence of an initial region partition can cause aliasing effects in the final output. For example, when the boundary of one of the square grids is close to, but not exactly at, an actual boundary in the image, the latter may never be recovered. This was not true of the last experiment where no artificial boundaries existed within the image. In addition, the effect of this phenomenon will diminish as the size of the grid approaches the resolution of the image, as was the

case with the first experiment. In general, the less the boundary conditions imposed by the initial partition, the smaller will be its effect on the final output.

Another notable aspect of the results, is that in both the first and last experiments, the final output was better (qualitatively and quantitatively) than the outputs obtained using merging or splitting rules individually. This can be confirmed by comparing the results in tables (7.1), (7.2), and (7.4).

In addition to the quality of the output, the efficiency of the computation is a second factor to consider when choosing the initial partition. In table (7.4), the number of actions and the number of cycles decreased steadily with the decrease in the number of initial regions. This was accompanied by an increase in the parameter F . Therefore, the computational factor favors the selection of an initial single region over a grid of fine regions, even if both lead to the same output quality.

7.2.4. Line Deletion Experiments

The next set of experiments tested the performance of rules that delete noisy lines from a line partition. Therefore, an initial line map was generated based on a low threshold on the gradient image, as shown figure (7.12a). This allowed for the detection of all the correct boundaries in the line map. The low threshold was also more sensitive

to small gradient variations. Hence, the initial map contained additional lines that did not correspond to actual boundaries.

Experiment 16 employed rule (1401) which deletes lines based on very low length and low average gradient. On the other hand, experiment 17 used rule (1402) for removing lines with very low gradient and low length. Rule (1403) was applied in experiment 18 to delete lines with low gradient and length (none had to be very low), with the additional constraint that the lines be isolated (with both ends open). In order to determine whether their effect was cumulative, the three rules were used together in experiment 19. The resulting line maps are shown in figures (7.12b) to (7.12e), and the performance measures are given in table (7.5).

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
	Initial	0	0.44	0.22	53.23	39.90	47.04	0.00
16	669	258	0.54	0.25	30.97	45.10	38.69	0.54
17	602	188	0.56	0.26	26.37	43.10	35.73	0.82
18	657	246	0.61	0.34	18.81	48.30	36.65	0.69
19	713	302	0.67	0.31	10.02	49.20	35.50	0.70
20	430	23	0.47	0.23	46.75	43.00	44.91	0.21
21	721	310	0.72	0.32	4.81	51.90	36.86	0.61
22	685	274	0.66	0.28	11.37	41.70	30.56	1.05

Table (7.5): Line deletion experiments.

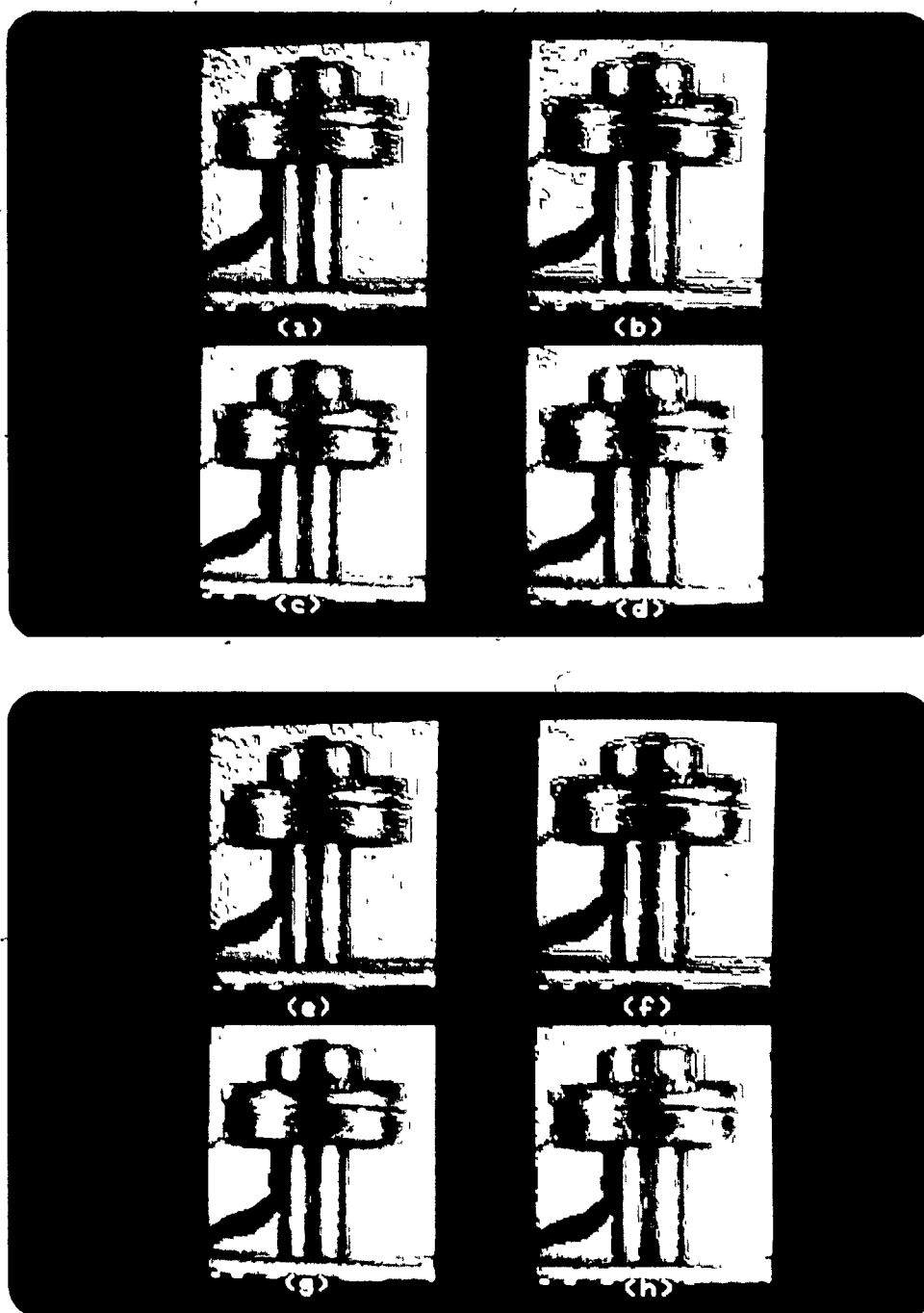
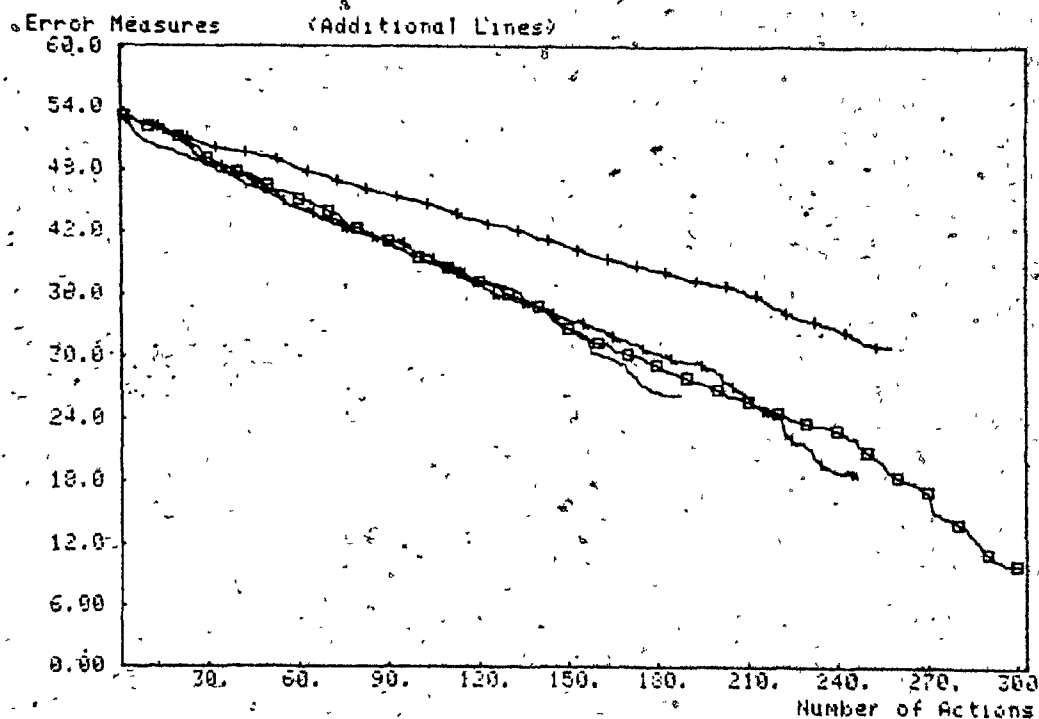


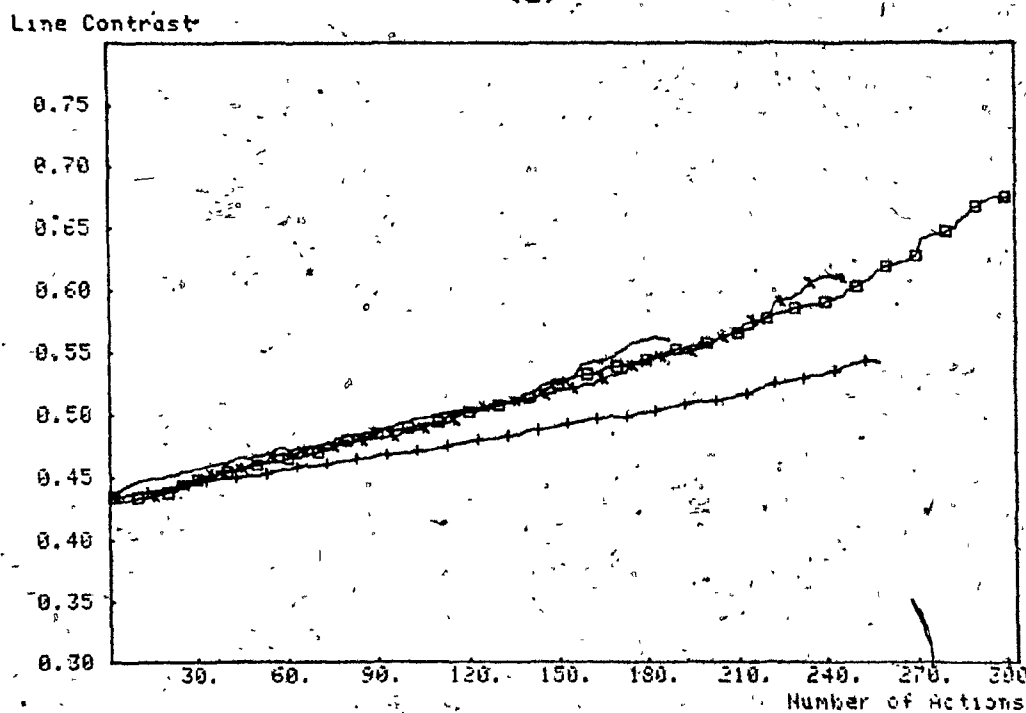
Figure (7.12): Line maps for deleting lines experiments:
 (a) Initial line map
 (b) Exp. 16: Deleting very short lines.
 (c) Exp. 17: Deleting lines with very low gradient
 (d) Exp. 18: Deleting short lines-with low gradient.
 (e) Exp. 19: Using the three criteria in (b, c and d)
 (f) Exp. 20: Deleting parallel lines only
 (g) Exp. 21: Combining (e) and (f)
 (h) Exp. 22: Deleting based on region information.

Each of the four experiments (16, 17, 18, and 19) resulted in a decrease in the additional lines error measure, as shown in figure (7.13a). The missing lines error increased slightly, but essentially remained constant, so that the total error curves more or less followed those in figure (7.13a). The contrast across lines also improved, as shown in figure (7.13b), while the line connectivity remained constant. It can be seen that the curve for rule (1401) is not as good as the rest. The curves for the other two rules behaved in an identical manner, with rule (1403) doing slightly better. The combined curve lead to a lower overall error and a better contrast measure, so that there is an evident gain in retaining both rules (1402) and (1403). A final remark on the curves in figures (7.13a) and (7.13b) is that they confirm our earlier assertion that the performance measurements (here the line contrast) behaved consistently with the segmentation error measures (in this case the additional lines error).

Experiment 20 examined the effect of applying rule (1404) to delete parallel lines that are multiple indications of a boundary. The rule was then included with the previous three line deletion rules in experiment 21 to determine if, in fact, it provided any additional knowledge. The curves for additional lines error for the two experiments were compared to that of experiment 19 when only the first three rules were used, as shown in figure (7.13c). The reduction in the error brought about by rule (1404) is



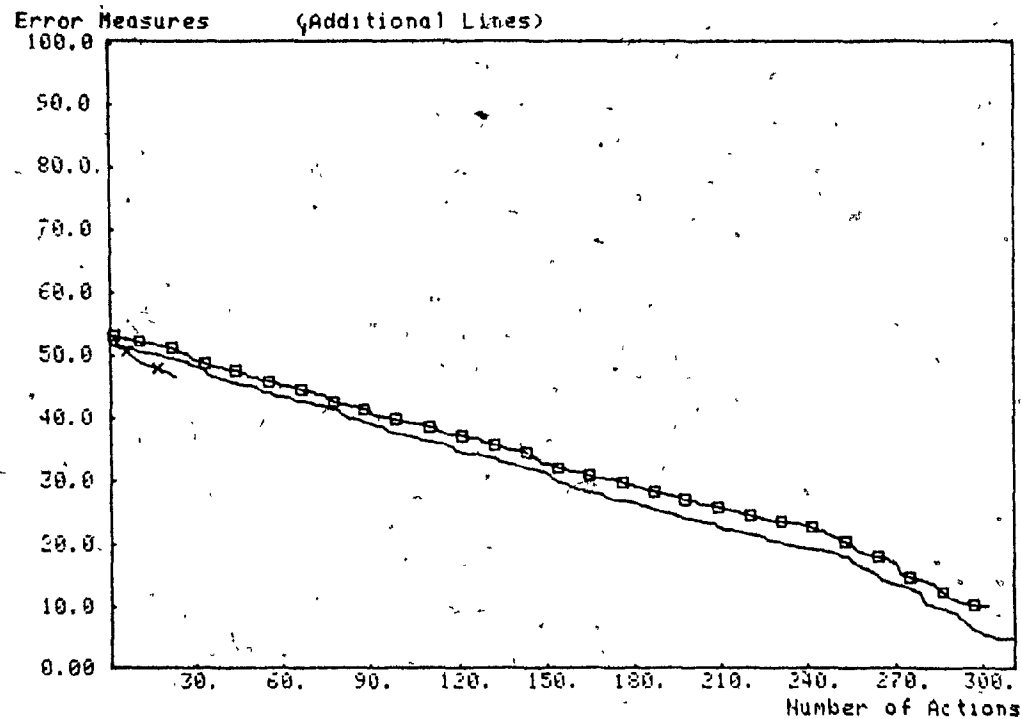
(a)



(b)

Figure (7.13): Performance measures for deleting lines.

Exp. 16 +
 Exp. 17 -
 Exp. 18 x
 Exp. 19 □



(c)

Figure (7.13) contd.: Exp. 19 □
 Exp. 20 ×
 Exp. 21 —

represented by the short curve. It can be seen that the difference between the other two curves (with and without the rule) is almost equal to the enhancement obtained when the rule was used by itself. This was also true of the line contrast curves for the three experiments. This is a clear indication that the knowledge embedded in rule (1404) supplements the other rules. The output line maps corresponding to experiments 20 and 21 are shown in figures (7.12f) and (7.12g), respectively.

The effect of including region information in rules for deleting lines was studied in experiment 22. Rules (1405) and (1406) deleted lines based on the similarity of regions on both sides. They were used in conjunction with the region map obtained through merging and splitting, as in experiment 15 above. The output is shown in figure (7.12h), and the performance measures are included in table (7.5). The additional lines error measure was compared to that of experiment 17, where rule (1402) deleted lines based on a very low average gradient. More noisy lines were removed when the region information was included, and this resulted in a additional 60% reduction in the error measure and a similar increase in the average line contrast. This provides evidence as to the importance of using regions and lines simultaneously in the analysis, a result that will find more support in other experiments that follow.

7.2.5. Line Extension Experiments

A different initial line map was used to test the rules that fill in gaps in a line partition. Therefore, the gradient image was thresholded at a higher value this time. This resulted in an initial line map with less noise, but more missing lines, as shown in figure (7.14a). Experiment 23 implemented rules (1507), (1508), (1607), and (1608) which extend lines in both directions based on local gradient. On the other hand, rules (1501) to (1503) and (1601) to (1603) extended lines based on their proximity to

neighboring lines in experiment 24. Both sets of rules were combined in experiment 25 to determine whether local gradient and proximity had an additive effect in detecting missing lines. Figures (7.14b) to (7.14d) show the output lines for the three experiments, and table (7.6) contains their performance measurements.

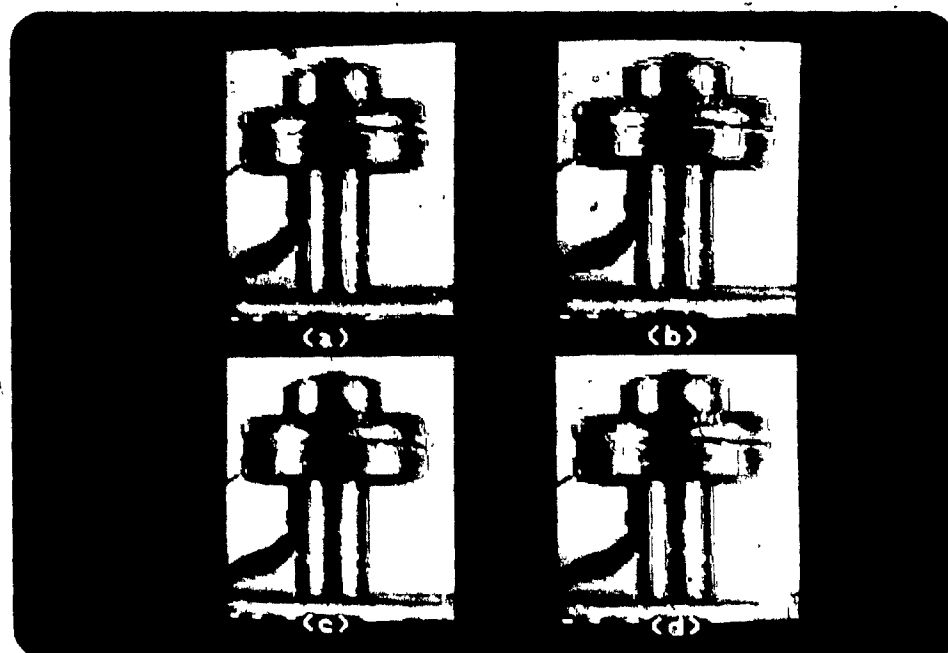


Figure (7.14). Line maps for extending lines experiments.
 (a) Initial line map
 (b) Exp. 23 : Extension based on local gradient
 (c) Exp. 24 : Extension based on proximity.
 (d) Exp. 25 : The combined criteria in (b) and (c).

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
	Initial	0	0.53	0.26	21.65	67.50	50.12	0.00
23	82	17	0.51	0.66	20.61	50.70	38.70	0.55
24	142	77	0.44	0.89	28.76	28.90	28.83	0.59
25	139	74	0.45	0.89	27.40	19.10	23.62	0.74
26	137	73	0.44	0.72	17.59	10.10	14.34	1.03

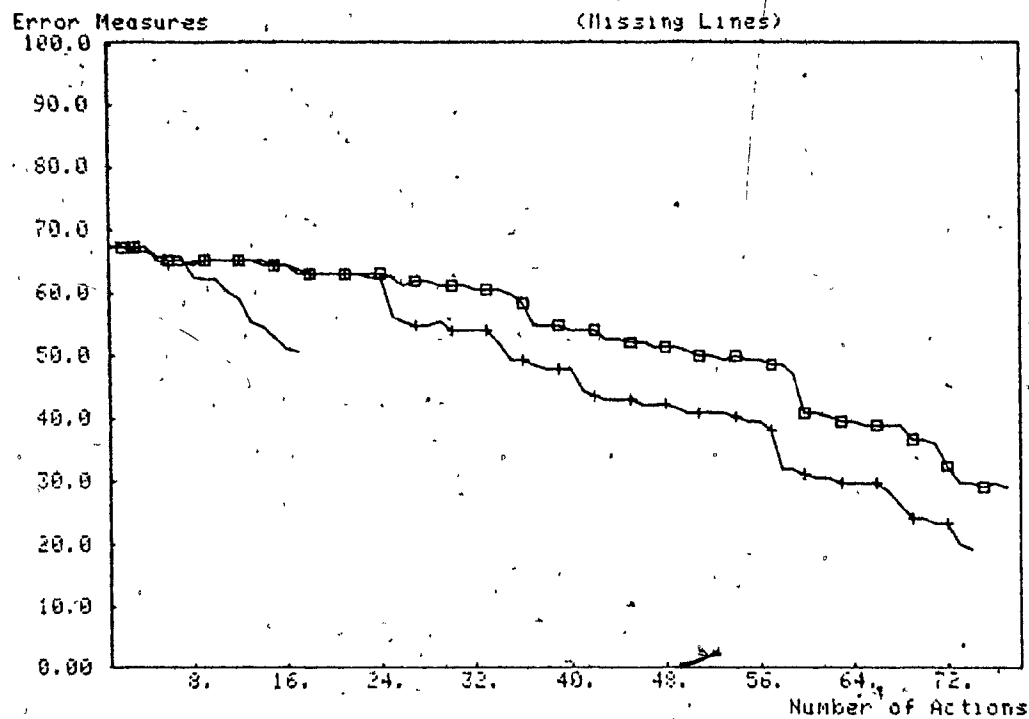
Table (7.6): Line extension experiments.

The local gradient method reduced the missing line error while keeping the additional line error constant. Proximity had more than double the effect on missing lines, but at the expense of slightly increasing the noisy lines. In contrast with the line deletion experiments, most of the changes affected the missing lines error measure shown in figure (7.15a), while the additional lines error measure remained almost unchanged. Similarly, it is the line connectivity that improved with these rules, as shown in figure (7.15b), whereas the line contrast decreased slightly. The curves show that the proximity method was superior overall, although there is obvious gain in including both sets of rules.

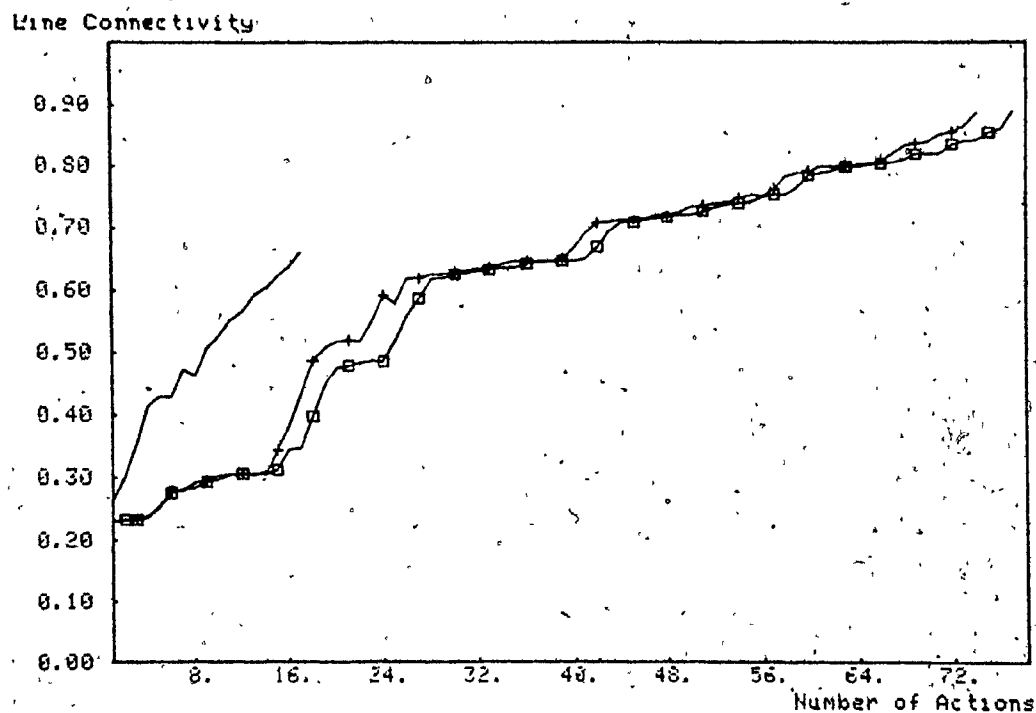
The effect of region information on line extensions was studied in experiment 26, when rules (1504) to (1506), (1509) to (1511), (1604) to (1606), and (1609) to (1611) were tested. The output is compared in figure (7.16) to experiment 25 where only line-based information was used. The missing lines error and the line connectivity curves are shown in figure (7.17). In both cases, the simultaneous use of regions with lines proved tangibly superior.

7.2.6. Line Merging Experiments

When two lines are merged together, the description of the new line differs from that of any of its constituents. However, the global structure of the line map will almost remain constant. Therefore, the segmentation error measures



(a)



(b)

Figure (7.15): Performance curves for extending lines.

Exp. 23 —

Exp. 24 □

Exp. 25 +

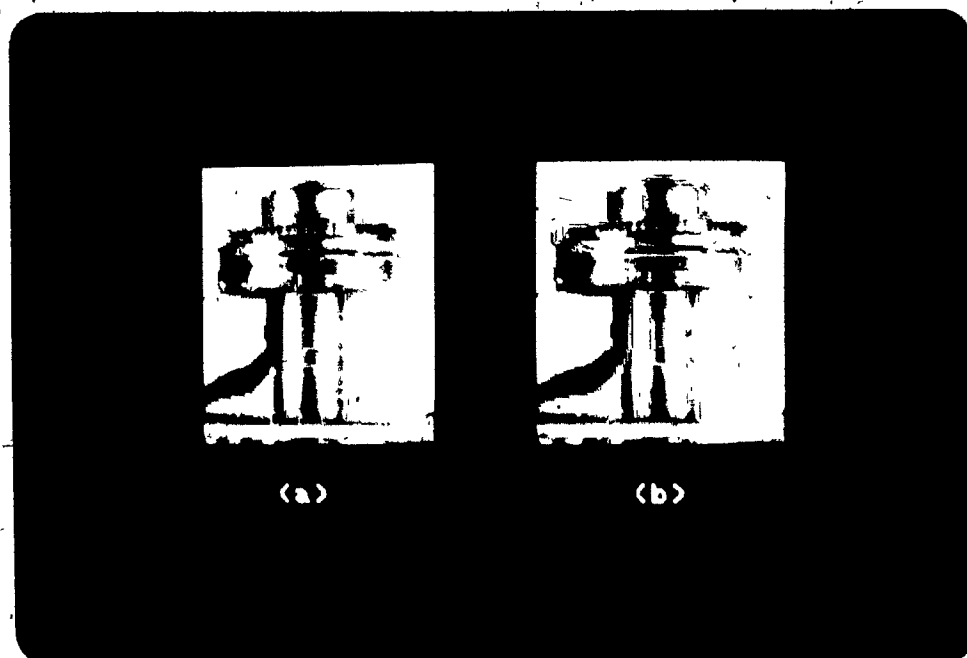
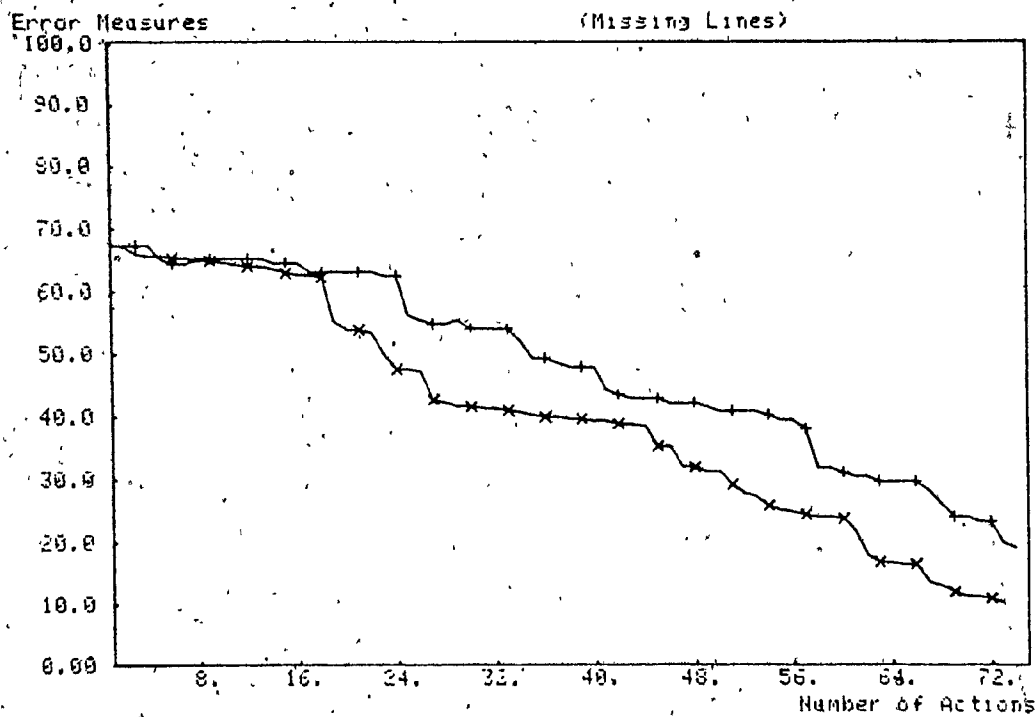


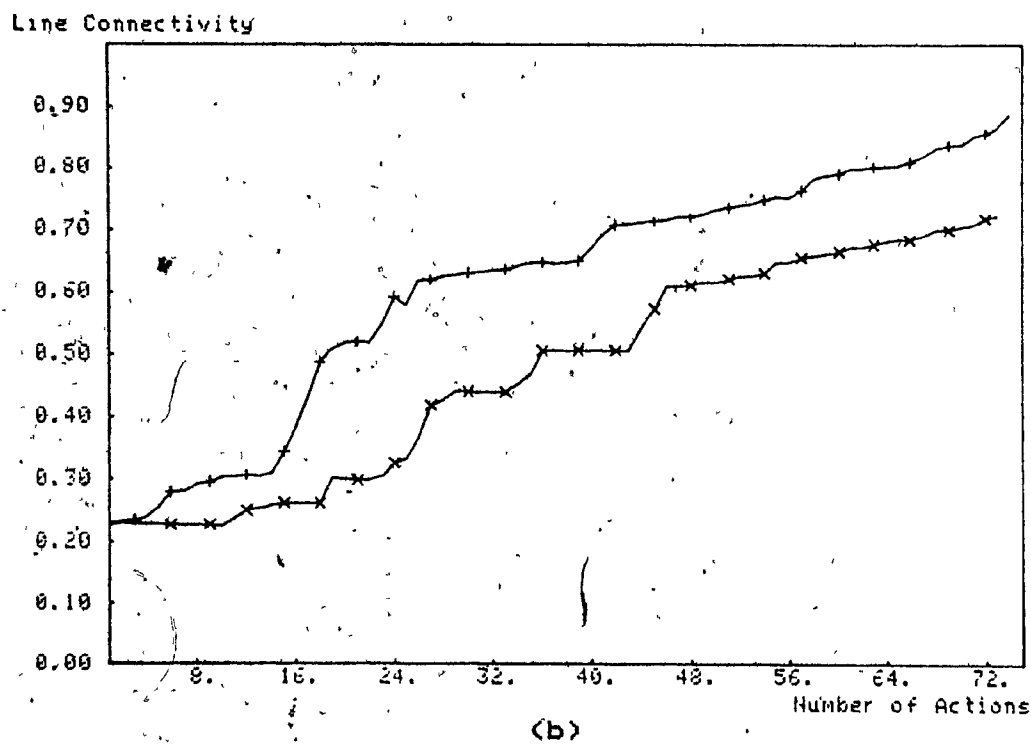
Figure (7.16): The effect of regions on line extensions.
 (a) Exp. 25 : Without Region information.
 (b) Exp. 26 : With region information.

for lines are less sensitive to this action, because the amount of additions and deletions to the lines involved are minimal. This made it more difficult to judge the performance of line merging rules based on the error measures. Nevertheless, comparisons based on the performance measures, such as line connectivity and the number of cycles were still possible.

Experiment 27 used rules (1701), (1702), (1801) and (1802) to merge lines based on proximity and similar directions. On the other hand, experiment 28 employed rules (1703) and (1803) for merging based on similarity of features. Both sets of rules were combined in experiment 29.



(a)



(b)

Figure (7.17): Performance curves for the effect of regions on line extension.

Exp. 25 +

Exp. 26 x

while in experiment 30 merging was based on continuity in the directions of both lines, according to rules (1704), (1705), (1804), and (1805)

Table (7.7) shows the resulting measures, where all experiments exhibited similar improvements in line connectivity, although the number of actions varied. There was a noticeable improvement in the additional lines error when continuity was considered, albeit at an added cost in the number of cycles. In general, this class of rules proved to be of little influence on the final output, as can be seen by superimposing the final (in green) and initial (in red) line maps for experiment 30 in figure (7.18). The yellow lines are those that did not change, red sections have been deleted and any additions would have appeared in green

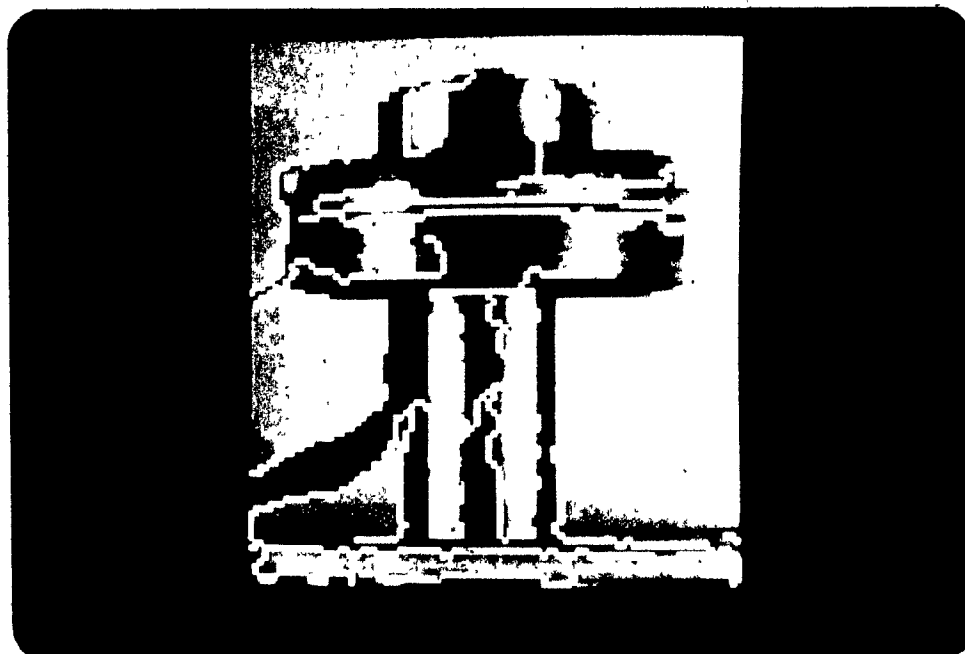


Figure (7.18): The connected line map superimposed on the initial (red sections have been deleted).

Exp.	Cycles	Actions	H	T	Additnl	Missing	Total	F
	Initial	0	0.51	0.88	26.83	49.00	39.50	0.00
27	84	11	0.58	0.98	20.53	49.10	37.63	0.86
28	83	12	0.52	0.96	17.99	49.20	37.04	1.15
29	93	23	0.53	0.98	16.94	49.40	36.93	1.07
30	129	39	0.48	0.97	12.95	49.50	36.18	1.00

Table (7.7): Line merging experiments.

Although they do not cause a significant change in the line map, the merging rules improve the structural description of individual lines. These lines become more representative of the physical boundaries in the image, as seen from the examples in figures (4.12) and (4.13). In addition, in some cases the merging action is only a first step. One of the resulting lines may be deleted by other rules, as demonstrated in figure (4.11c). Neighboring regions may also benefit from the merging of two lines. The new line configuration could improve the region partition by guiding subsequent merging or splitting actions.

7.2.7. The Effect of Lines on Regions

In the previous sections, rules that use region information in line analysis have been shown to substantially improve the results. The next experiments tested rules that work in the opposite direction, reconciling the output region partition with information obtained from the lines.

Experiment 31 performed region analysis using rules (906) and (907) as the best splitting rules selected in section 7.2.2. However, the best merging rule (801) was substituted by rule (809) which restricts merging if a line exists between the regions. In addition, line analysis employed the best line deletion, extension, and merging rules selected in the previous sections. The initial region map contained one region. The results were compared to experiment 15, which used rules (906), (907), and (801) on the same initial map (i.e., without line analysis). The outputs for 15 and 31 are shown in figures (7.19a) and (7.19b), respectively, and their measures are summarized in table (7.8). The new merging rule kept the under-merging error measure at a lower level, although this was offset by a similar increase in the over-merging error. Both experiments had equivalent total error measures. However, when line analysis was included, the result was achieved in less than half the actions, and one quarter the matching cycles. This improvement is reflected in the values of F for the two experiments.

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
	Initial	0	0.67	0.54	15.25	37.30	41.93	0.00
15	468	111	0.89	0.80	26.34	25.00	25.68	0.74
31	112	48	0.89	0.49	17.13	34.90	27.49	2.77
32	250	88	0.90	0.56	22.96	20.90	21.95	1.71
33	191	67	0.79	0.56	14.72	24.40	20.15	2.45

Table (7.8): Effect of lines on regions.

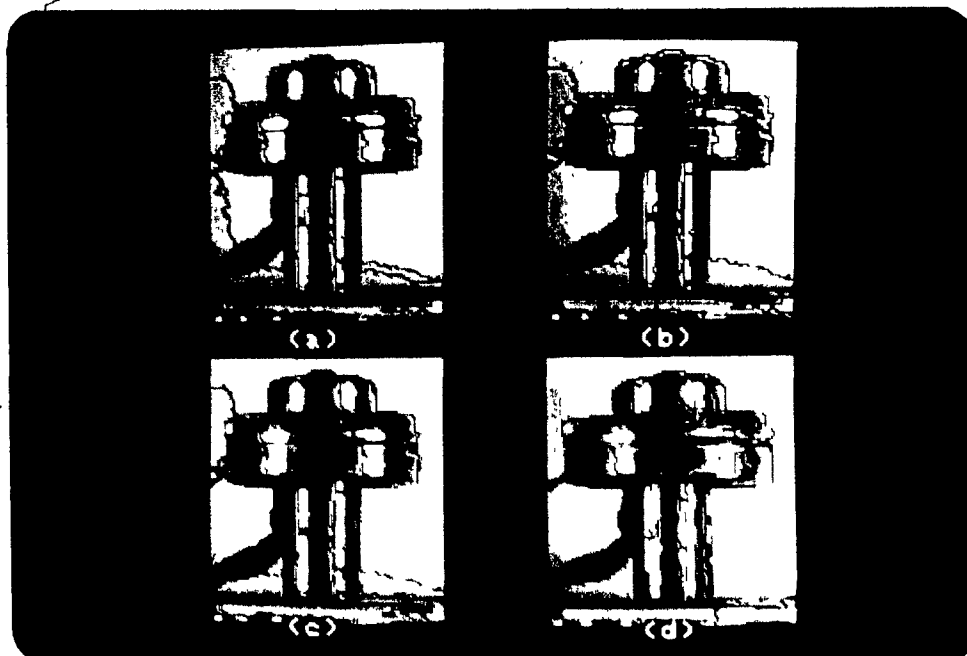


Figure (7.19): The effect of using line information on the region maps.

- (a) Exp. 16 : No line information.
- (b) Exp. 31 : Line information in merging only.
- (c) Exp. 32 : Lines used for both merging and splitting (with priority to histogram splitting).
- (d) Exp. 33 : Same as (c) except with priority to splitting on lines.

Experiment 32 extended the argument one step further by adding rule (908) which splits a region at existing lines. This implicated line information in both merging and splitting. The results exhibited an improvement in the overall error measure and an increase in the number of cycles, as shown in table (7.8).

In the previous experiment the histogram splitting rules (906) and (907) were given priority over the line splitting rule (908). Experiment 33 reversed the priorities.

This resulted in a slight improvement in the error, as well as a decrease in the number of cycles. This was because the initial line map was fairly developed as shown in figure (7.12a) with respect to the initial single region map which contained virtually no information. Consequently, a benefit was gained in this case by allowing the information from the lines to propagate faster to the regions. These strategy considerations and others will be emphasized in section 7.3. The output maps for 32 and 33 are shown in figure (7.19c) and (7.19d), respectively, and the total error curves for the four experiments are given in figure (7.20). Note the improvement in the final error level and in the speed of reaching that level as line information was gradually introduced into the region analysis.

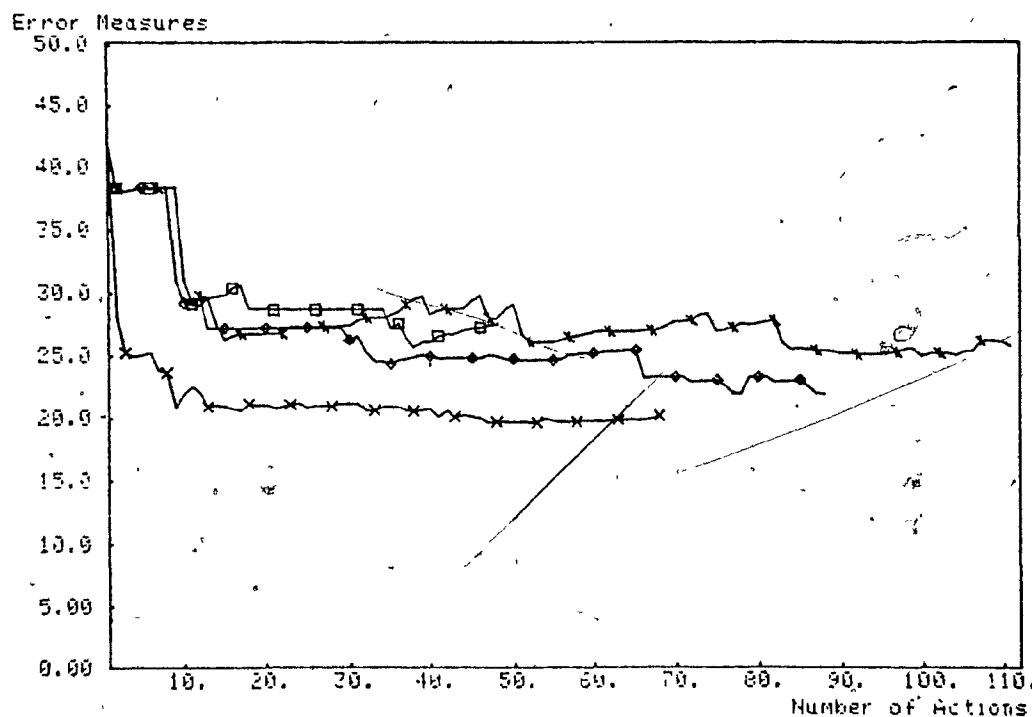


Figure (7.20): The effect of lines on region analysis as reflected in the total error measure.

Exp. 15 ● Exp. 32 ◇
Exp. 31 □ Exp. 33 ×

7.2.8. Knowledge Rules Evaluation

The experiments conducted in the previous sections studied the relative importance of the processing criteria that are coded in the rules. By examining and comparing the measurement tables, the region and line map outputs, and the performance curves, a number of conclusions were made as to the effectiveness and efficiency of different rule combinations. In addition to these, an individual evaluation of every rule was made throughout these experiments, as well as other experiments that used a variety of input pictures. Whenever a rule fired, the changes in the performance measurements due to that rule's action were accumulated in the rule statistics relation of equation (3.34), together with a count of the total number of times the rule fired relative to the number of times it was tested (defined as the hit ratio). Based on these, a ranking of all the rules was computed and stored in the rule priority relation of equation (3.35), as described in section 3.5.2. The results are presented in table (7.9).

An optimal set of twenty-two rules was created by selecting the highest ranking rules from each rule category. In some cases more than one rule was selected, namely, where the experiments have shown a substantial additive effect of two or more rules. The set consists of: rule (809) for region merging; rules (906), (907), and (908) for region splitting; rules (1402), (1404), (1405), and (1406) for line deletion; rules (1503), (1507), (1508), (1510) and (1511)

Category	Effectiveness	Activity	Rule Number
8	VERY HIGH	HIGH	809 *
8	HIGH	VERY HIGH	801
8	MEDIUM	MEDIUM	802
8	MEDIUM	MEDIUM	806
8	MEDIUM	HIGH	808
8	LOW	VERY LOW	803
8	LOW	LOW	807
8	VERY LOW	VERY LOW	804
8	VERY LOW	VERY LOW	805
9	VERY HIGH	MEDIUM	908 *
9	HIGH	MEDIUM	906 *
9	HIGH	HIGH	907 *
9	MEDIUM	LOW	905
9	MEDIUM	MEDIUM	904
9	MEDIUM	HIGH	903
9	LOW	HIGH	902
9	LOW	VERY HIGH	901
14	VERY HIGH	LOW	1404 *
14	VERY HIGH	HIGH	1405 *
14	HIGH	MEDIUM	1406 *
14	HIGH	HIGH	1402 *
14	MEDIUM	MEDIUM	1401
14	MEDIUM	HIGH	1403
15	VERY HIGH	LOW	1507 *
15	VERY HIGH	MEDIUM	1510 *
15	HIGH	LOW	1508 *
15	HIGH	MEDIUM	1511 *
15	HIGH	HIGH	1503 *
15	MEDIUM	VERY LOW	1505
15	MEDIUM	VERY LOW	1506
15	MEDIUM	LOW	1509
15	MEDIUM	MEDIUM	1504
15	MEDIUM	HIGH	1501
15	LOW	HIGH	1502
16	VERY HIGH	LOW	1607 *
16	VERY HIGH	HIGH	1610 *
16	HIGH	LOW	1611 *
16	HIGH	MEDIUM	1608 *
16	HIGH	HIGH	1603 *
16	MEDIUM	VERY LOW	1606
16	MEDIUM	LOW	1605
16	MEDIUM	LOW	1609
16	MEDIUM	MEDIUM	1604
16	MEDIUM	HIGH	1601
16	LOW	HIGH	1602

Table (7,9): Rule ranking (continued on next page).
Rules in the optimal set are denoted by *.

17	MEDIUM	LOW	1704 *
17	MEDIUM	MEDIUM	1705 *
17	LOW	VERY LOW	1701
17	LOW	LOW	1703
17	VERY LOW	LOW	1702
18	MEDIUM	VERY LOW	1804 *
18	MEDIUM	MEDIUM	1805 *
18	LOW	LOW	1801
18	LOW	LOW	1803
18	VERY LOW	LOW	1802

Table (7.9) contd.: Rule ranking.
Rules in the optimal set are denoted by *.

for extending lines forward; rules (1603), (1607), (1608), (1610) and (1611) for extending lines backward; and finally rules (1704), (1705), (1804), and (1805) for merging lines. This rule set was used in the following experiments to determine system behaviour under different processing strategies.

7.3. CONTROL STRATEGY EXPERIMENTS

The basic idea behind the dynamic strategy mechanism introduced in chapter 6 is that better performance can be obtained if the system control adapts to varying data conditions. The experiments presented in this section illustrate how the different strategy elements introduced in section 6.2 affect the analysis. The strategy elements were summarized in table (6.1). Each was shown to assume one of a specific number of states.

A set of experiments was conducted for each of the six elements of strategy. Within each set, different experiments

recorded system performance when one strategy element was assigned each of its states. The segmentation error measures and the performance measures used in the previous section were computed under various data conditions. A comparison between the results obtained for all the states lead to conclusions about the best state (strategy setting) to use for different ranges of performance parameters. These conclusions correspond to the constraints introduced in tables (6.2) and (6.3). They control dynamic strategy selection by defining the relation between performance measures and strategy setting.

Sections 7.3.1. to 7.3.6. describe the experiments conducted for each strategy element. A brief summation of the resulting constraints is presented in the form of a table at the end of each section. The experiments were conducted on the image of figure (7.21a), and the error measures were computed with respect to the hand segmented region map of figure (7.21b).

7.3.1: Region Path Strategy Experiments

Four states were tested that differed in the method by which they select the next region to visit: by highest adjacency value, by largest size, by smallest size, and by lowest adjacency value. Experiments 34 to 37 employed the four path strategies to analyze the image when the initial region map was a single region. Therefore, the region features initially had low uniformity and high contrast. The

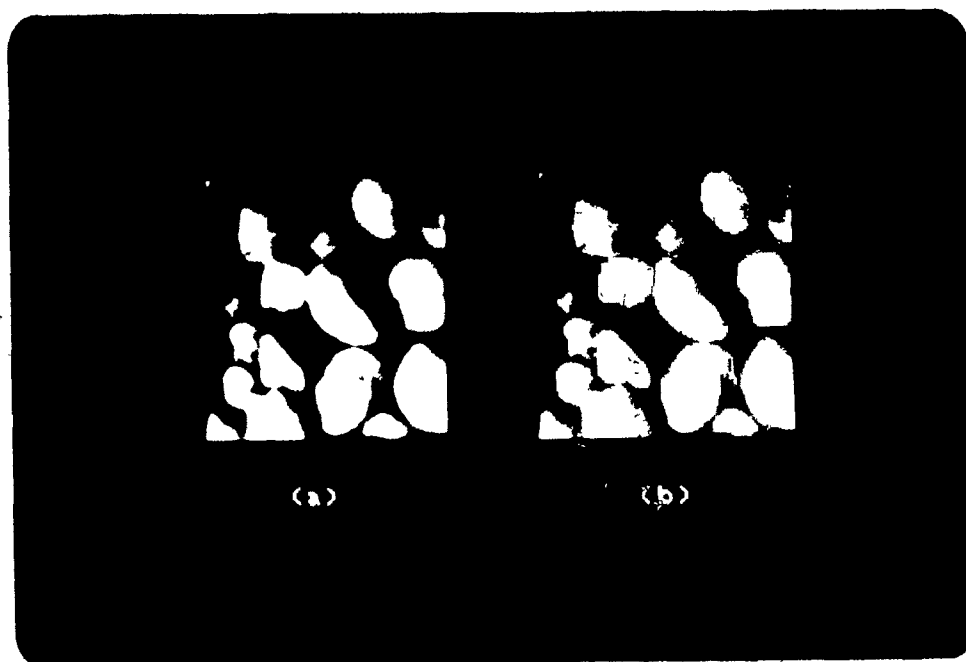


Figure (7.21): Test image for strategy experiments
 (a) Original image.
 (b) Hand segmentation.

number of regions increased gradually with processing and hence their average size decreased. The resulting region maps are shown in figures (7.22a) to (7.22d), and the performance measures are summarized in table (7.10a). The lowest overall segmentation error was obtained for experiment 35 when larger regions were considered first. This was mainly due to a lower under-merging error brought about by producing the smallest number of regions. The same experiment resulted in the largest over-merging error, as evident from the grouping of the two rocks in the center of the image. Experiment 37 (lowest adjacency) yielded the largest number of regions, and hence the highest under-merging error and the lowest average contrast.

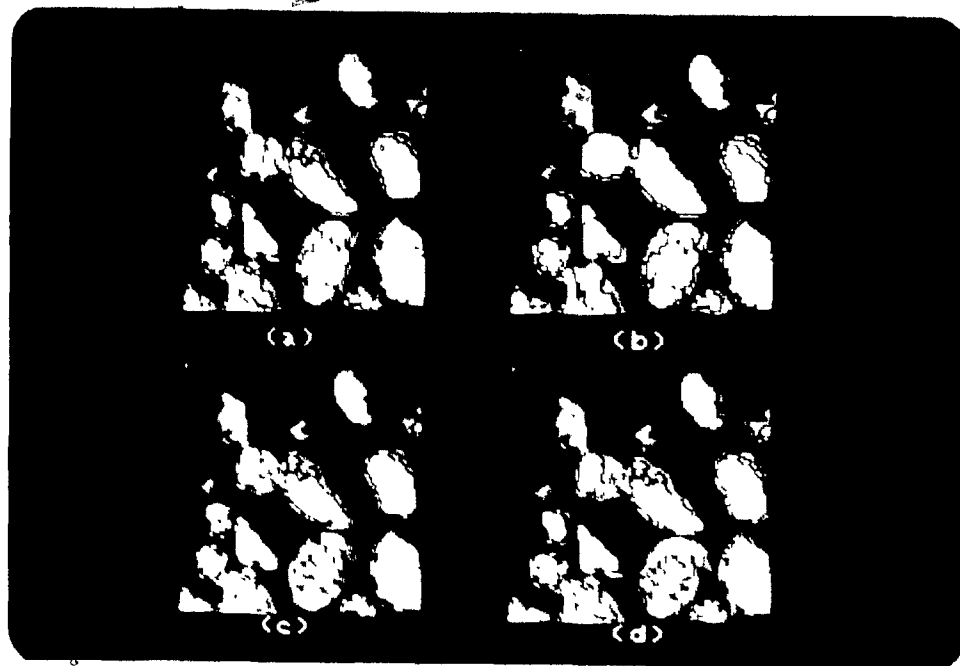


Figure (7.22): First set of region maps for path strategies.
 (a) Exp. 34 : Highest adjacency.
 (b) Exp. 35 : Largest size.
 (c) Exp. 36 : Smallest size.
 (d) Exp. 37 : Lowest adjacency.

Region Uniformity

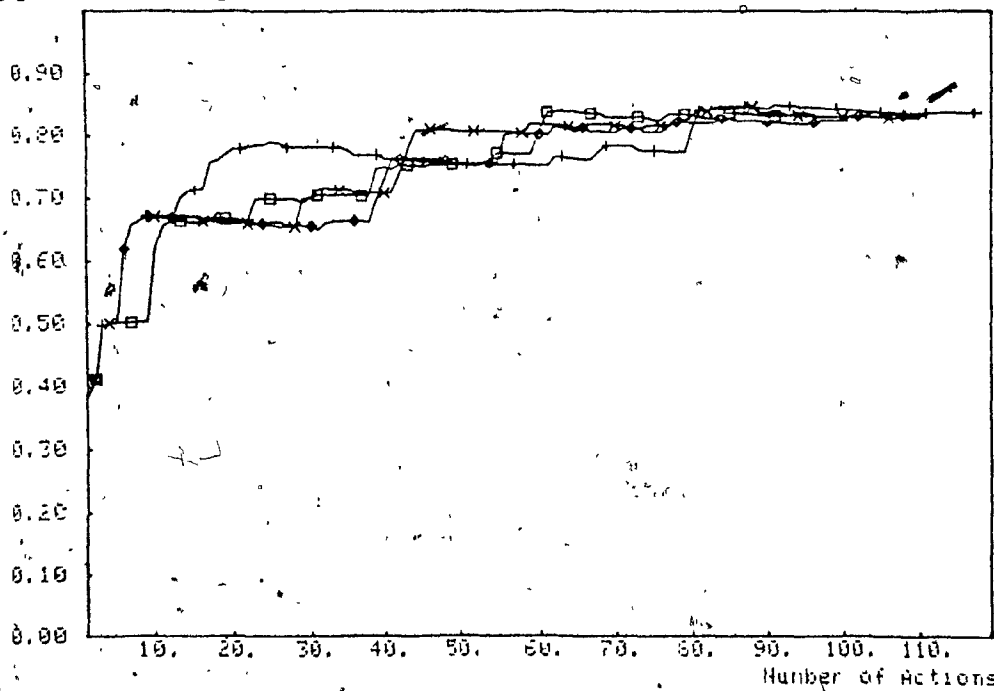


Figure (7.23): Region uniformity for region path exps.
 Exp. 34 x Exp. 36 □
 Exp. 35 + Exp. 37 ◇

In the previous experiments processing started with a single region. Therefore, the most significant changes occurred in the region uniformity measure, which was initially very low. Figure (7.23) shows the curves for uniformity as a function of the number of actions. In the early stages, the strategy, which favored larger size regions was superior. Later on, high adjacency produced the best performance. Finally, when the number of regions became large enough, selecting the smallest regions resulted in the highest measures. By imposing the proper constraints, a dynamic strategy will in effect select the best curve for each range of uniformity values (or number of regions). In this case, very low uniformity and very few regions favored the path that selected larger regions. High adjacency was preferred for high uniformity with a relatively small number of regions.

A second set of experiments tested the effects of having a large number of initial regions with high uniformity and low contrast. The initial region map was composed of 2X2 square grid that contained 4200 regions. The performance measurements for the four path strategies are given in table (7.10b) for experiments 38 to 41 in the same order as above. The output maps are shown in figures (7.24a) to (7.24d). In this case, the lowest overall error and the highest average contrast were due to the strategy that favored smaller regions. The average contrast curves of figure (7.25) indicate that giving priority to smaller

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
Initial	0	0	0.38	0.84	0.00	71.70	50.70	0.00
34	305	110	0.83	0.94	17.53	9.90	14.24	1.17
35	318	118	0.84	0.92	13.72	11.00	12.44	1.18
36	342	96	0.83	0.94	21.73	10.40	17.03	0.97
37	377	111	0.83	0.91	18.45	8.70	14.42	0.94

(a)

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
Initial	0	0	0.90	0.47	94.12	4.80	66.64	0.00
38	3250	2313	0.79	0.77	46.57	9.30	33.58	0.89
39	3215	2290	0.80	0.76	48.10	9.20	34.63	0.87
40	2978	2271	0.77	0.81	40.59	10.10	29.58	1.09
41	2814	2269	0.77	0.72	46.05	10.90	33.46	1.03

(b)

Table (7.10): Region path strategy experiments.

(a) Initial number of regions = 1

(b) Initial number of regions = 4200

regions is more beneficial when the number of regions is relatively high. This conforms with the results of the first set of experiments, except that the motivating measure in this case is the contrast (as opposed to uniformity before). Table (7.11) lists the resulting constraints.

To summarize, contrast is more important when the number of regions is high, and it favors the processing of small regions. Uniformity is the dominant factor when few regions are present, and the selection of large regions becomes more

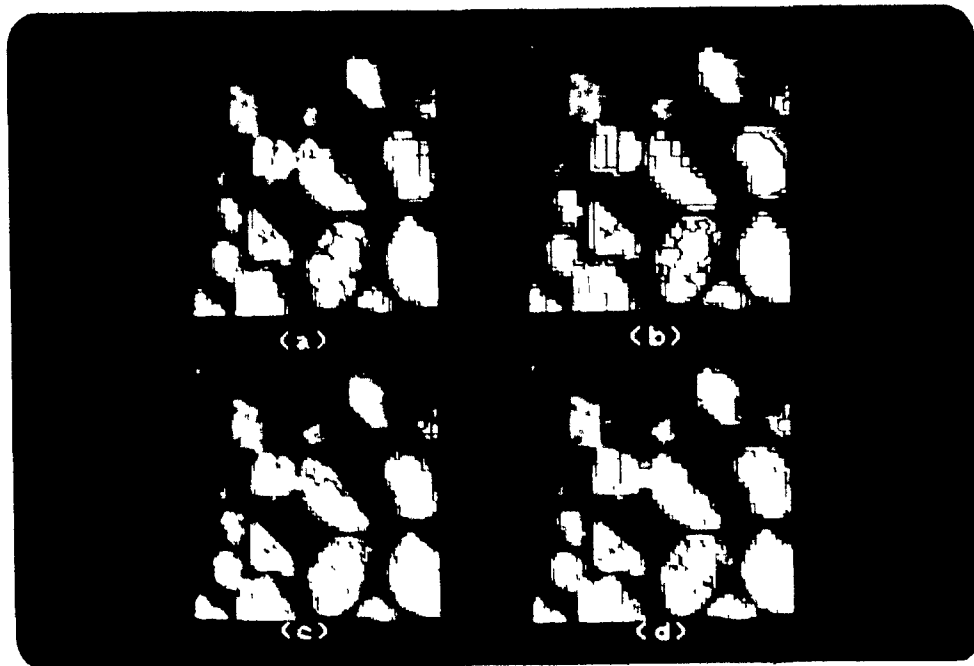


Figure (7.24): Second set of region maps for path strategies.

- (a) Exp. 38 : Highest adjacency.
- (b) Exp. 39 : Largest size.
- (c) Exp. 40 : Smallest size.
- (d) Exp. 41 : Lowest adjacency.

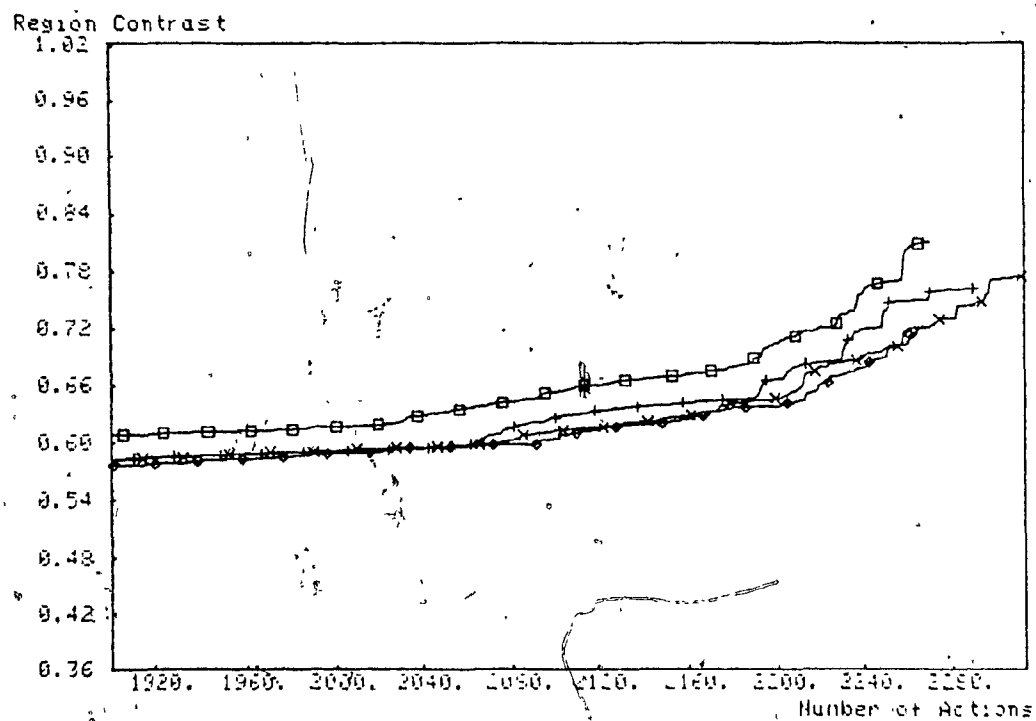


Figure (7.25): Region contrast for region-path exps.

- Exp. 38 x
- Exp. 39 +
- Exp. 40 □
- Exp. 41 ◇

	LOW	HIGH
NUMBER OF REGIONS	LARGEST	SMALLEST
REGION UNIFORMITY	SMALLEST	ADJACENT
REGION CONTRAST	LARGEST	ADJACENT

Table (7.11): Relation between performance measures and region path strategy.

preferable. At intermediate values, higher adjacency as a criterion for path selection dominates the transition from one extreme to the other. Low adjacency is not a decisive factor in selecting the processing path.

7.3.2. Line Path Strategy Experiments

The five methods that were tested for selecting the next line in the processing path are: the nearest, the longest, the highest gradient, the shortest, and the lowest gradient. Two initial line maps were used to reflect the varying data conditions. The first contained fewer lines that had a high average contrast and low connectivity, as shown in figure (7.26a). The second was produced by using a lower threshold on the gradient array. Therefore, it contained a larger number of lines, with lower average contrast, and an

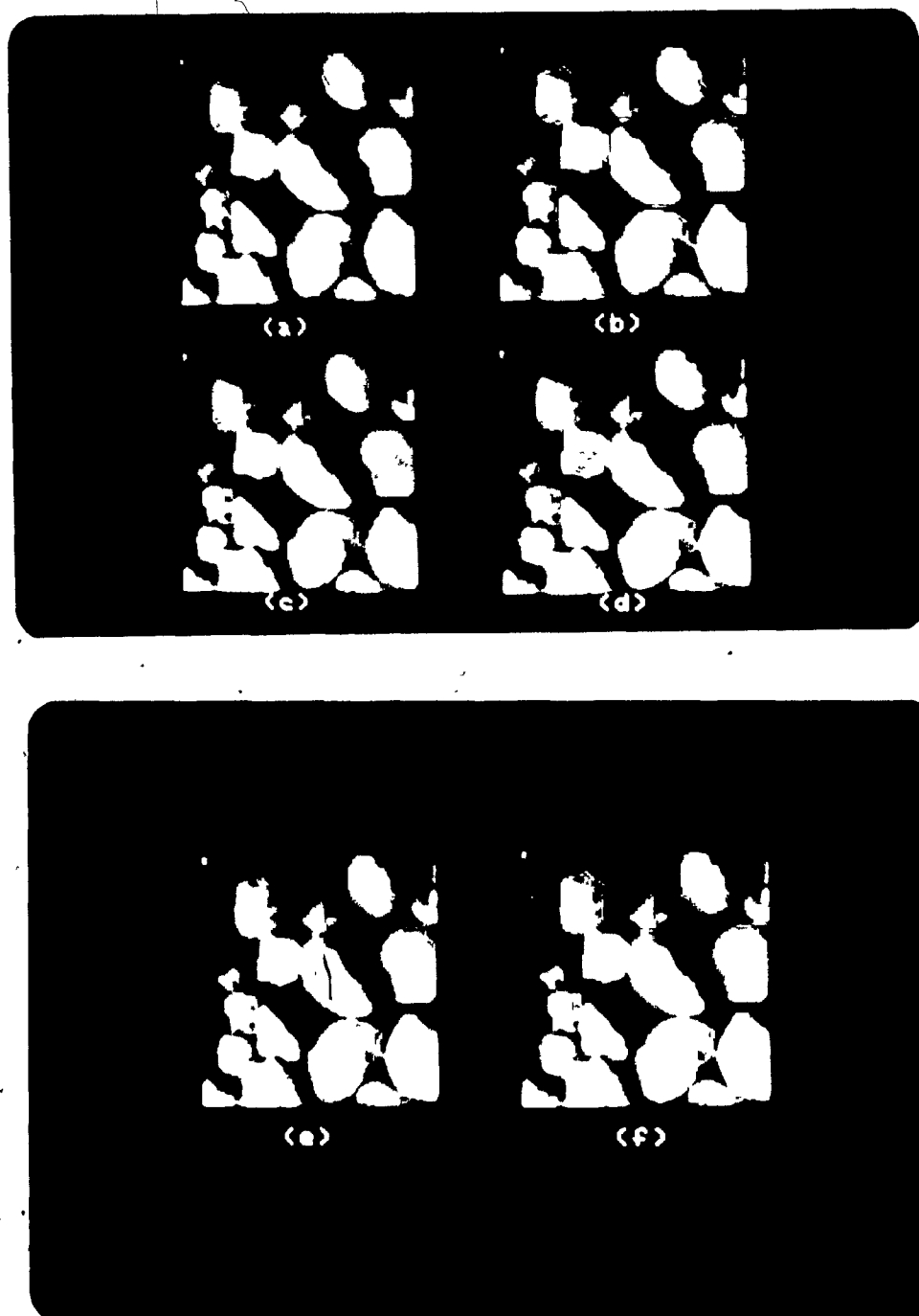


Figure (7.26) First set of line maps for line path strategy

(a)	Initial map with low number of lines
(b) Exp. 42	Nearest line.
(c) Exp. 43	Longest.
(d) Exp. 44	Strongest (highest gradient).
(e) Exp. 45	Shortest.
(f) Exp. 46	Weakest (lowest gradient).

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
	Initial	0	0.73	0.30	30.44	61.70	48.65	0.00
42	274	155	0.64	0.82	22.39	33.10	28.26	0.79
43	270	157	0.64	0.85	20.28	23.90	22.16	1.05
44	272	160	0.65	0.92	22.36	26.92	24.75	0.94
45	284	170	0.65	0.94	20.84	40.90	32.46	0.61
46	282	168	0.64	0.98	18.23	42.90	32.96	0.59

(a)

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
	Initial	1	0.54	0.30	55.19	43.50	49.69	0.00
47	487	264	0.71	0.76	21.73	37.10	30.40	0.81
48	472	255	0.72	0.75	12.71	36.50	27.33	0.97
49	468	251	0.72	0.89	14.19	37.10	28.09	0.95
50	555	275	0.72	0.81	7.11	31.90	23.11	0.98
51	540	269	0.75	0.89	3.91	31.90	22.73	1.02

(b)

Table (7.12): Line path strategy experiments.

(a) Initial number of lines = 94

(b) Initial number of lines = 178

equally low line connectivity value, as shown in figure (7.27a).

Experiments 42 to 46 applied the five path selection methods to the first line map, and the results are displayed in figures (7.26b) to (7.26f). Table (7.12a) shows the performance measures for these experiments. When the initial contrast was high, choosing the longer and stronger lines

produced better overall error measures and better line partitions. This is in contrast with the results of experiments 47 to 51 shown in table (7.12b) and figures (7.27b) to (7.27f). The selection of the shorter and weaker lines produced better error measures for the large number of lines with relatively low average contrast. Therefore, for the dynamic selection of the path, longer lines should be preferred over shorter ones, and stronger over weaker, when the average contrast is high, and vice-versa.

Other constraints are obtained by studying the line connectivity curves for the two sets of experiments shown in figure (7.28). Different curves produce the highest measure at different stages in the analysis. In (a), where the number of initial lines was small, priority to stronger lines (O) was better at low connectivity values. There was also a slight preference for longer lines (x). Both effects were reversed at higher connectivity where selecting weaker (+), and then shorter (/) lines became superior. The curves in (b), where the number of lines was initially higher, lead to different conclusions. Longer lines (x) were better at low connectivity and stronger lines (O) did better at high connectivity. These constraints are summarized in table (7.13).

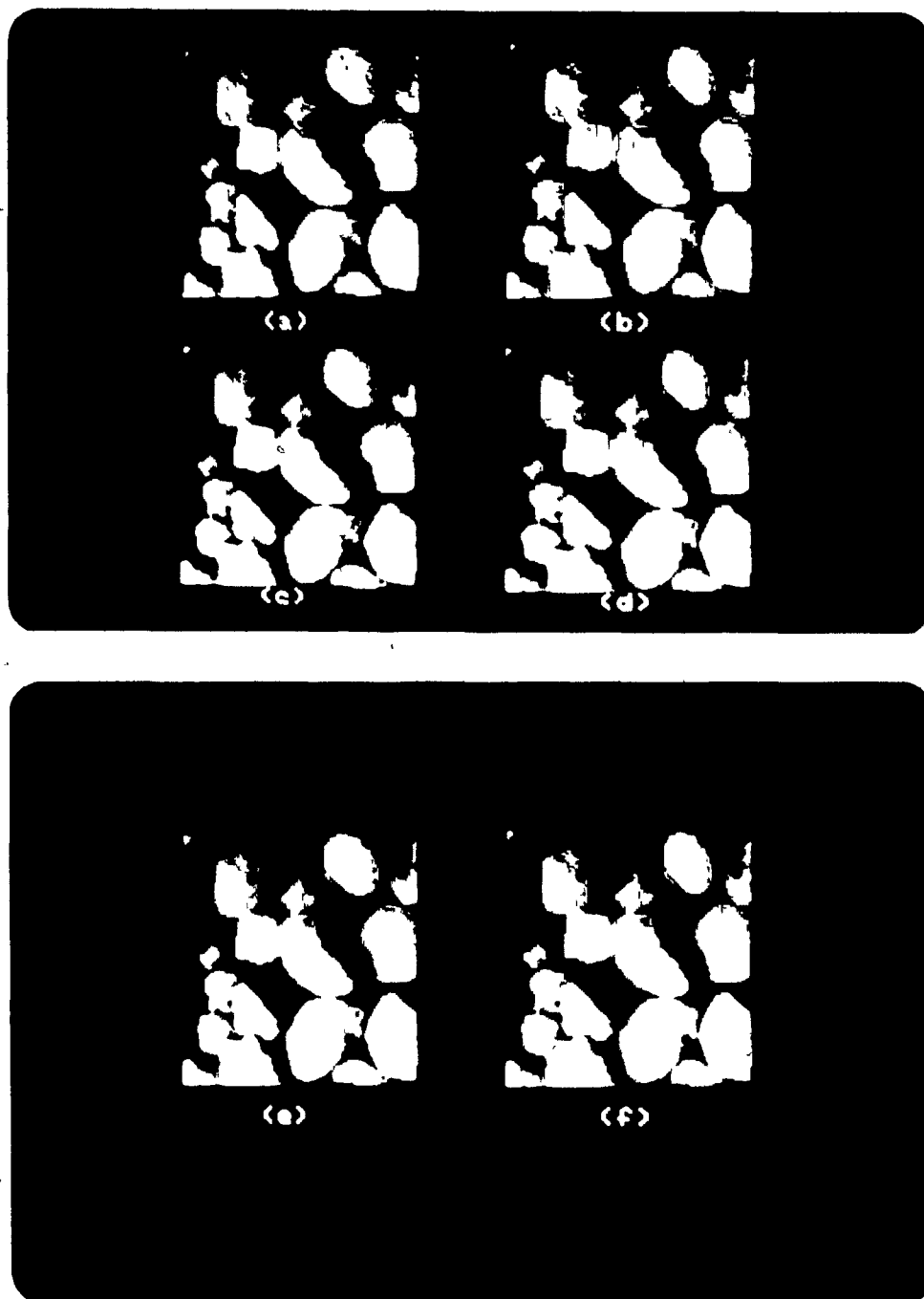
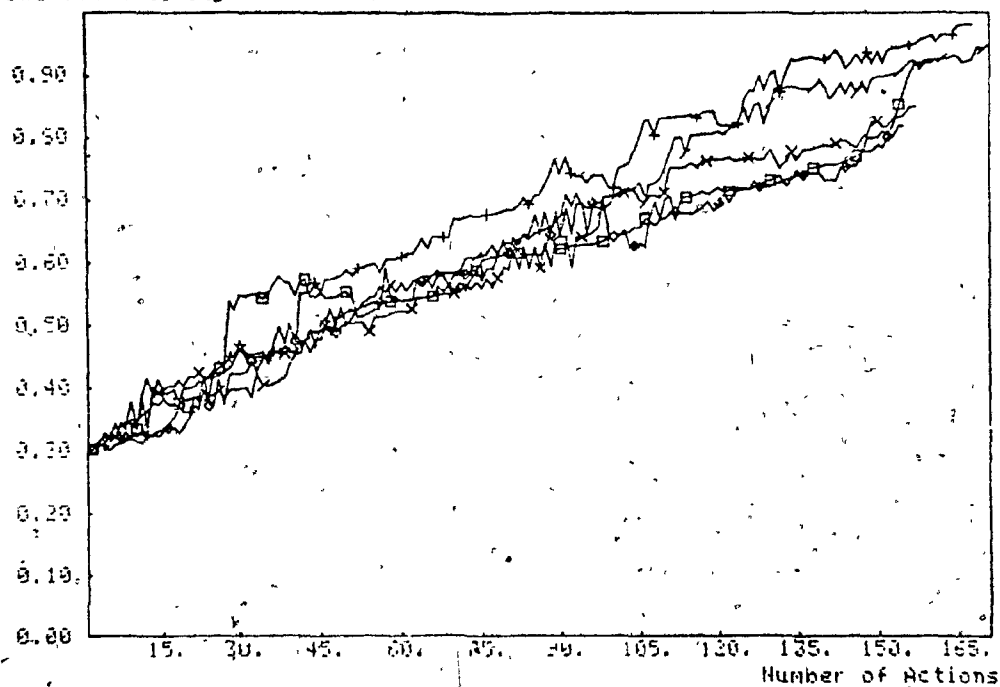


Figure (7.27): Second set of line maps for line path strategy

- (a) Initial map with high number of lines.
- (b) Exp. 47 : Nearest line.
- (c) Exp. 48 : Longest.
- (d) Exp. 49 : Strongest (highest gradient).
- (e) Exp. 50 : Shortest.
- (f) Exp. 51 : Weakest (lowest gradient).

Line Connectivity

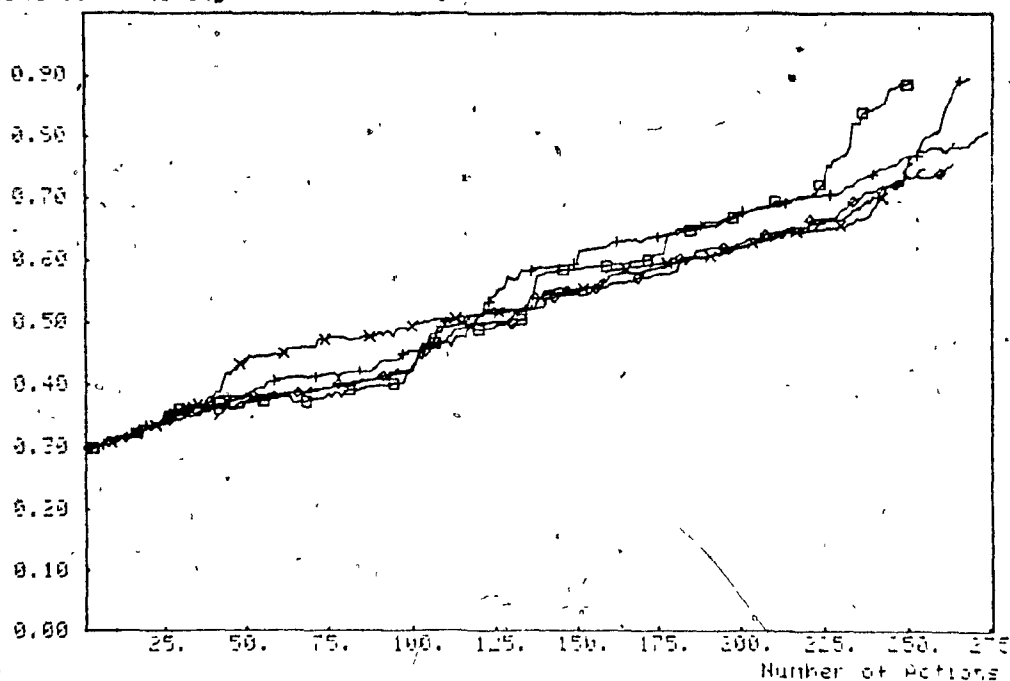


(a) Exp. 42 $\diamond$
Exp. 45 $/$

Exp. 43 $\times$
Exp. 46 $+$

Exp. 44 $\square$

Line Connectivity



(b) Exp. 47 $\diamond$
Exp. 50 $/$

Exp. 48 $\times$
Exp. 51 $+$

Exp. 49 $\square$

Figure (7.28): Connectivity curves for line path exps.
(a) A small number of initial lines.
(b) A large number of initial lines.

	LOW	HIGH
NUMBER OF LINES	LONGER	SHORTER
LINE CONTRAST	SHORTER	LONGER
LINE CONNECTIVITY	STRONGER	WEAKER

Table (7.13): The relation between performance measures and line path strategy.

7.3.3. Updating Strategy Experiments

The two states of this strategy component are obtained by either successively updating a region by repeatedly applying rules, or by moving on to other regions after a single update. Starting with the 2X2 grid of 4200 regions, single and successive updating were applied in experiments 52 and 53, respectively. The output segmentations are shown in figure (7.29a) and (7.29b), and the performance measurements are given in table (7.14a). It is seen that the more conservative single updating strategy resulted in a much lower over-merging error and a slightly higher under-merging error. Both produced regions of comparable uniformity, but better contrast was obtained using single updating. For example, in figure (7.29b) the darker rock at the top center was merged with the background. Figure (7.30a) shows how the

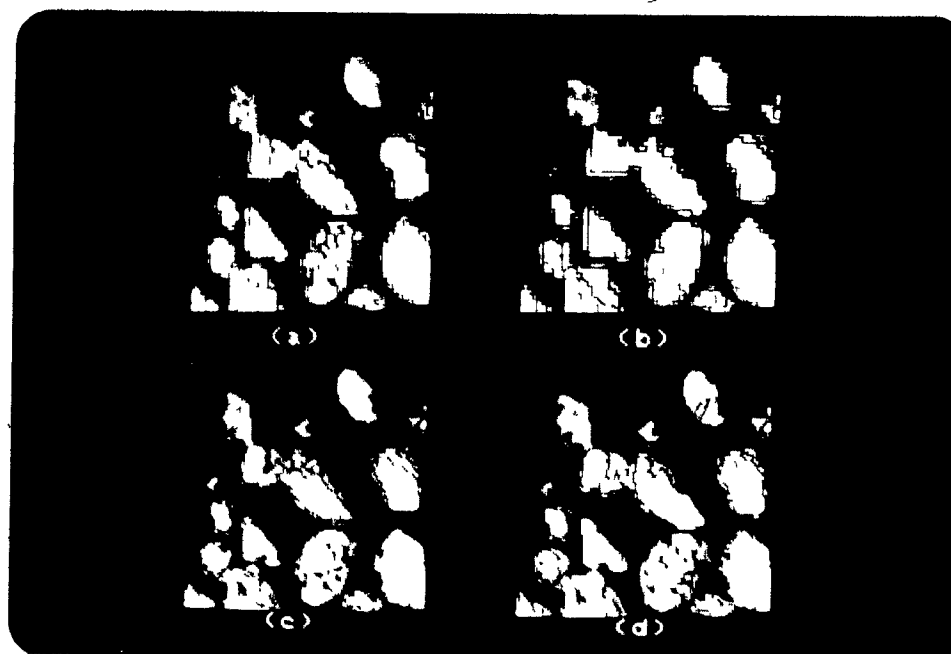


Figure (7 29). Region maps for region updating strategy.

- (a) Exp 52 Single updating starting with 4200 regions
 (b) Exp 53 Successive updating with 4200 regions
 (c) Exp 54 Single updating starting with one region
 (d) Exp 55 Successive updating with one region

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
	Initial	0	0.90	0.47	94.12	4.80	66.64	0.00
52	4531	2320	0.80	0.84	38.05	9.80	27.78	1.00
53	3600	2276	0.78	0.70	34.22	22.60	29.00	1.21

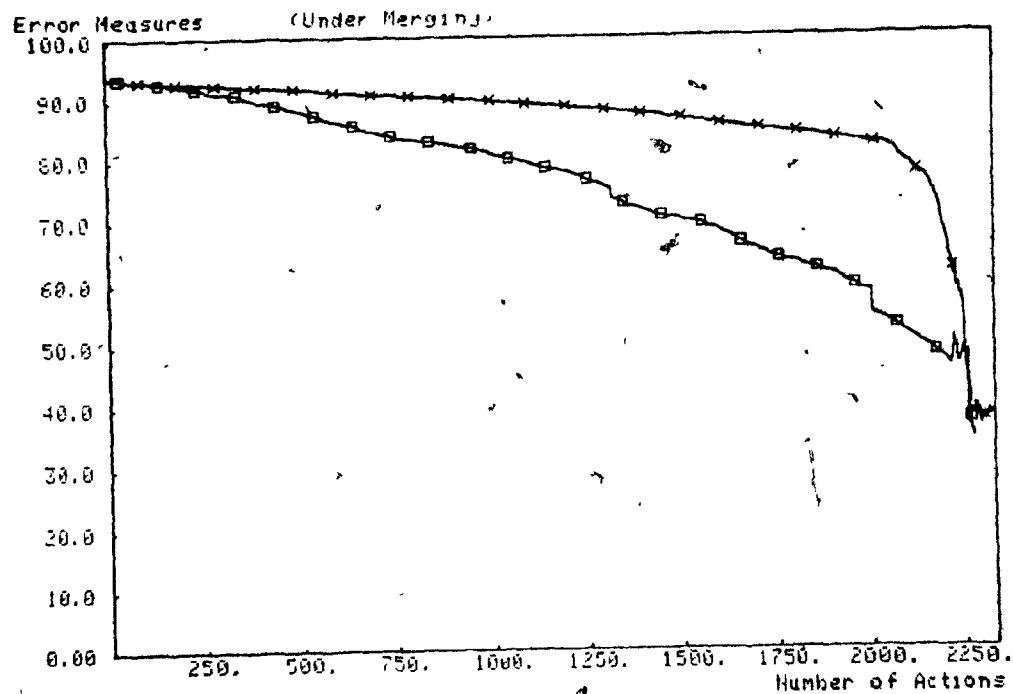
(a)

Exp	Cycles	Actions	U	C	Under	Over	Total	F
	Initial	0	0.38	0.84	0.00	71.70	50.70	0.00
54	405	88	0.84	0.92	18.25	9.60	14.58	0.99
55	269	93	0.84	0.91	17.54	10.40	14.42	1.50

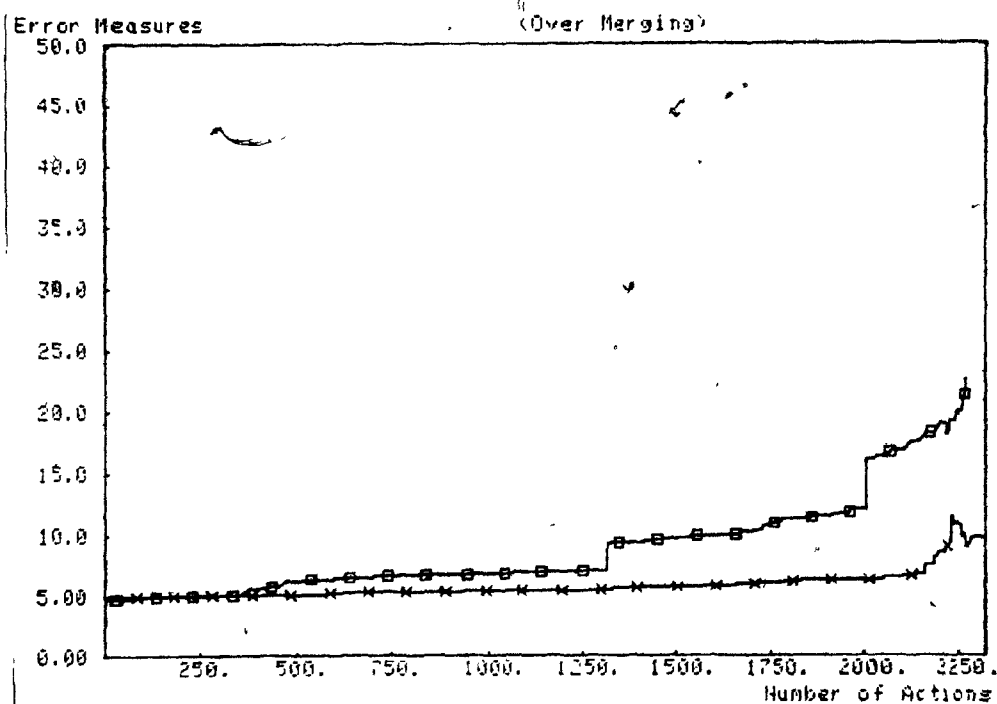
(b)

Table (7 14) Updating strategy for regions experiments.

- (a) Initial number of regions = 4200
 (b) Initial number of regions = 1



(a)



(b)

Figure (7.30): Error curves for region updating strategy.
 Exp. 52 x Exp. 53 o

under-merging error was reduced by each of the two strategies. Successive updating caused a faster improvement because the initial grid was rapidly merged to form large regions. On the other hand, single updating merged neighboring grid elements to form larger grids that grew slowly into larger regions. When the latter were merged near the end of processing, the error dropped significantly to almost the same level obtained through successive updating. However, figure (7.30b) shows how successive updating caused a substantially higher over-merging error all along the processing path. This justifies the larger number of cycles required by the single updating strategy.

The same two experiments were repeated when the initial map contained a single region. Single and successive updating were employed in experiments 54 and 55, respectively. The results are tallied in table (7.14b) and the output maps are displayed in figures (7.29c) and (7.29d). Because there were fewer regions, the two strategies produced similar results. This makes successive updating much more attractive in this case because it requires a much smaller number of cycles to produce the same output. In general, when fewer regions exist, even with low uniformity values, region splitting will perform adequately with successive updating. Conversely, a large number of regions with low average contrast requires repeated merging, and this is best done using a single updating strategy.

The two states of the updating strategy were tested again for two initial line maps. Experiments 56 and 57 employed single and successive strategies, respectively, for the initial line map of figure (7.26a), which contained fewer lines of high contrast and low connectivity. Figure (7.31a) and (7.31b) show the resulting line maps, while table (7.15a) compares their performance measurements. For a small number of lines with high initial contrast, successive updating produced a lower overall error measure for a comparable number of cycles.

Finally, single and successive updating were tested in experiments 58 and 59 for the initial map of figure (7.27a) with a larger number of lines and lower initial contrast. Figures (7.31c) and (7.31d) represent the output lines and table (7.15b) summarizes the final performance measures. This time single updating resulted in lower segmentation error, although it used almost twice the number of cycles.

The final value of the line connectivity was better in single updating for the two initial maps. A closer look at the line connectivity curves for experiments 58 and 59 in figure (7.32) shows that the improvement occurred at higher connectivity values. Since the curves are closer at low connectivity, successive updating is preferable because of its time saving capabilities.

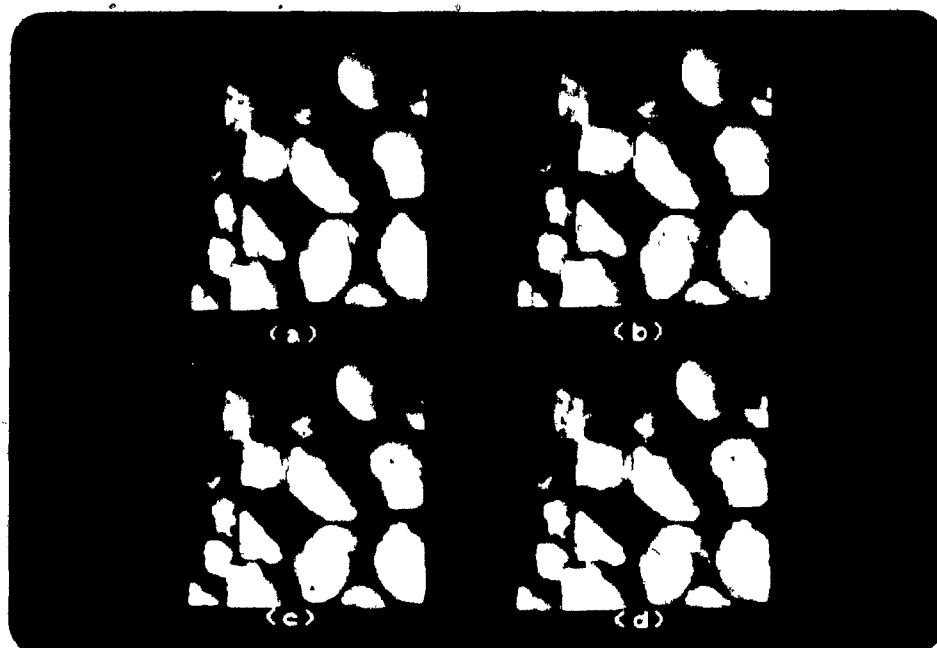


Figure (7.31): Line maps for line updating strategy.

- (a) Exp. 56: Single updating with low number of lines.
 (b) Exp. 57: Successive updating with low number of lines.
 (c) Exp. 58: Single updating with high number of lines.
 (d) Exp. 59: Successive updating with high number of lines

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
	Initial	0	0.73	0.30	30.44	61.70	48.65	0.00
56	218	152	0.62	0.94	25.80	40.30	33.84	0.65
57	207	141	0.64	0.81	22.70	29.00	26.04	1.05

(a)

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
	Initial	0	0.54	0.30	55.19	43.50	49.69	0.00
58	489	263	0.74	0.95	17.75	30.90	25.20	1.00
59	258	212	0.74	0.75	22.94	36.70	30.60	1.47

(b)

Table (7.15): Updating strategy for lines experiments.

(a) Initial number of lines = 94

(b) Initial number of lines = 178

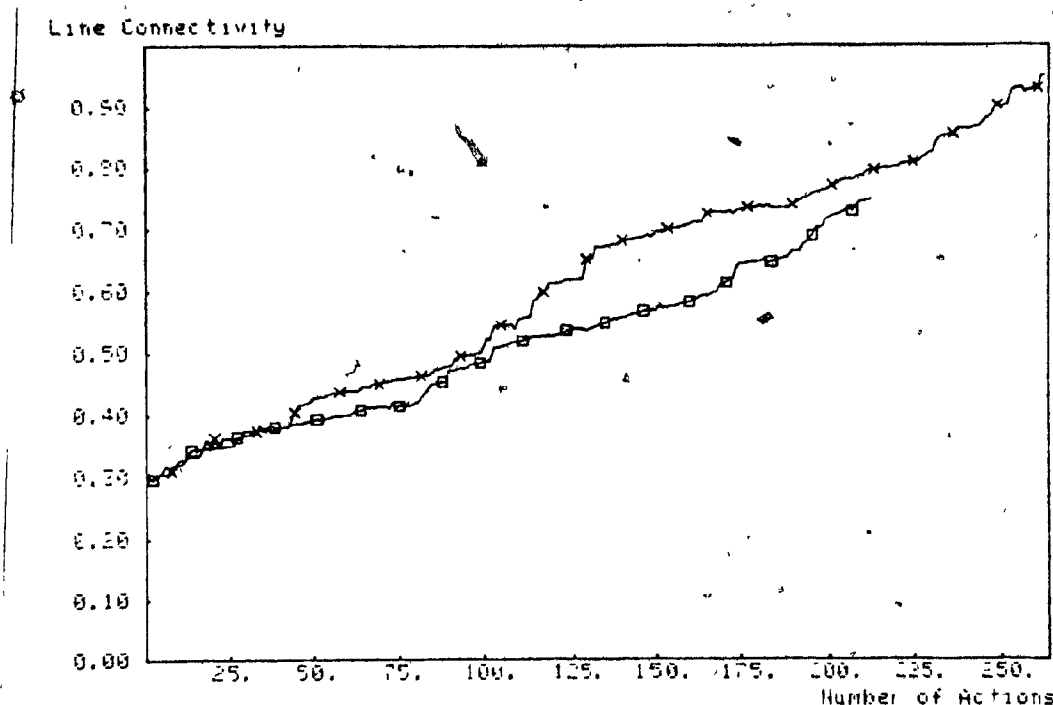


Figure (7.32): Connectivity curves for line updating strategy.
Exp. 58 x Exp. 59 □

Therefore, successive updating is more beneficial when the contrast is high, when the connectivity is low, and when fewer lines exist. On the other hand, low contrast, high connectivity and a larger number of lines would all favor a single updating strategy. The constraints for updating regions and lines are summarized in table (7.16).

7.3.4. Process Priority Experiments

The effects of assigning higher priorities to rules that use region information over those that use line information were tested in experiment 60. Line rules were given the edge in experiment 61 which used the same initial partitions. The results are reflected in the final region partitions shown in figure (7.33) and the measures of table (7.17).

	LOW	HIGH
NO OF REGIONS	SUCCESSIVE	SINGLE
NO OF LINES	SUCCESSIVE	SINGLE
REGION UNIFORMITY	SINGLE	SUCCESSIVE
REGION CONTRAST	SINGLE	SUCCESSIVE
LINE CONTRAST	SINGLE	SUCCESSIVE
LINE CONNECTIVITY	SUCCESSIVE	SINGLE

Table (7.16) The relation between performance measures and the updating strategy

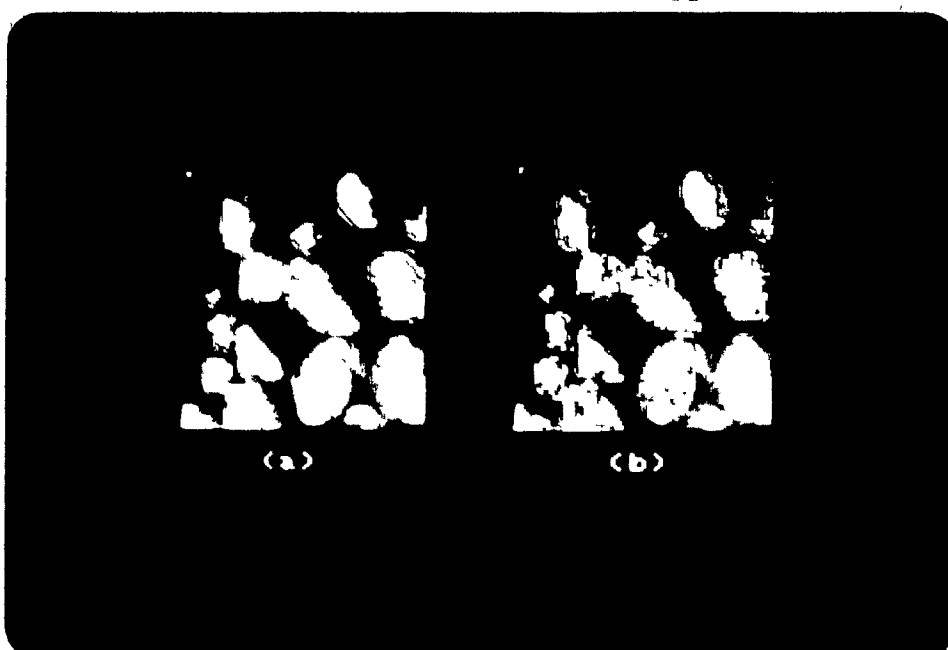


Figure (7.33). Region maps for process priority experiments
 (a) Exp 60: Priority for region rules
 (b) Exp 61: Priority for line rules

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
Initial	0	0	0.38	0.84	0.00	71.70	50.70	0.00
60	311	59	0.65	0.89	15.46	11.30	13.54	1.30
61	405	88	0.84	0.92	18.25	9.60	14.58	0.97

Table (7.17) Process priority experiments

Starting with the whole image as one region, the initial region uniformity was very low while the contrast was high. Both experiments increased the uniformity albeit in different ways, as shown in figure (7.34). At low uniformity values, assigning higher priorities to region analysis rules produced better results. The effect was reversed at high uniformity values.

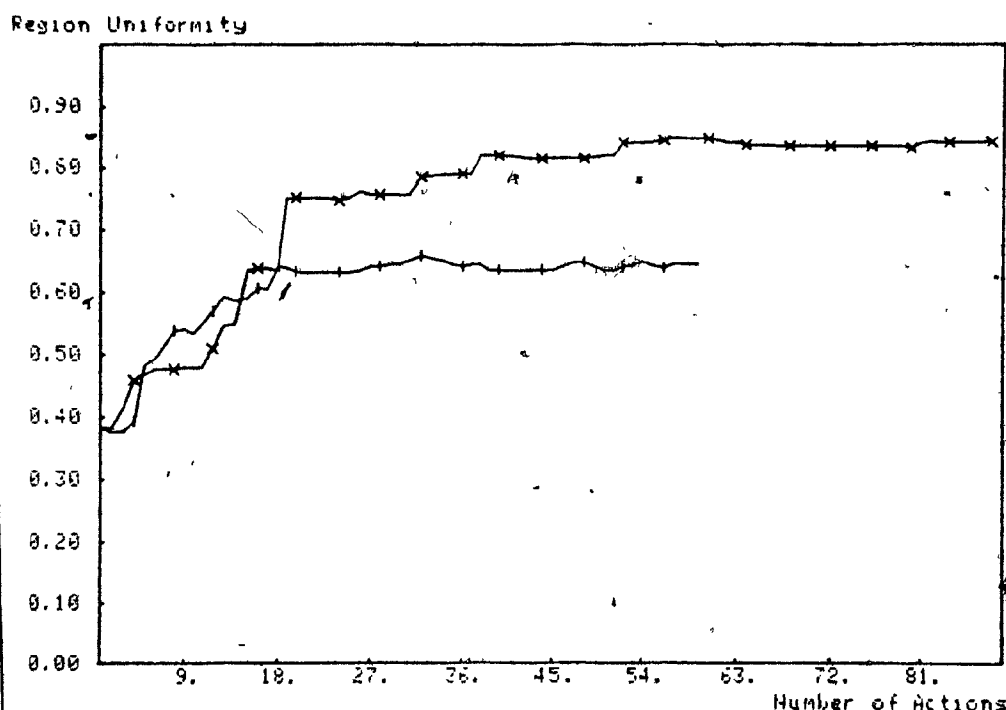


Figure (7.34): Region uniformity curves for process priority.
Exp. 60 + Exp. 61 x

Other experiments have shown that the above effect is true of all the other performance measures. For example, identical results were obtained when the initial region map had low contrast values. The effects were reversed when the region measures were initially high and the line measures were low. Therefore, the general constraint can be

summarized as follows: the process with lower performance measures should have rules with higher priority. Table (7.18) explicitly lists the individual constraints

	LOW	HIGH
REGION UNIFORMITY	REGIONS	LINES
REGION CONTRAST	REGIONS	LINES
LINE CONTRAST	LINES	REGIONS
LINE CONNECTIVITY	LINES	REGIONS

Table (7.18): The relation between performance measures and process priority.

7.3.5. Region Rules Priority Experiments

Starting with the single region map, experiment 62 assigned higher priority to merging rules, while in experiment 63 splitting rules were matched first. Figure (7.35) and table (7.19) show that the final results were very close. However, because more splitting was needed for this initial map, the curves in figure (7.36) show that region uniformity improved faster when priority was given to the splitting rules.

A similar experiment using an initial region map with a large number of regions having high uniformity and low contrast yielded complementary results. The contrast improved faster when merging rules had higher priorities. The constraints that guide the selection of region rule priorities are summarized in table (7.20).

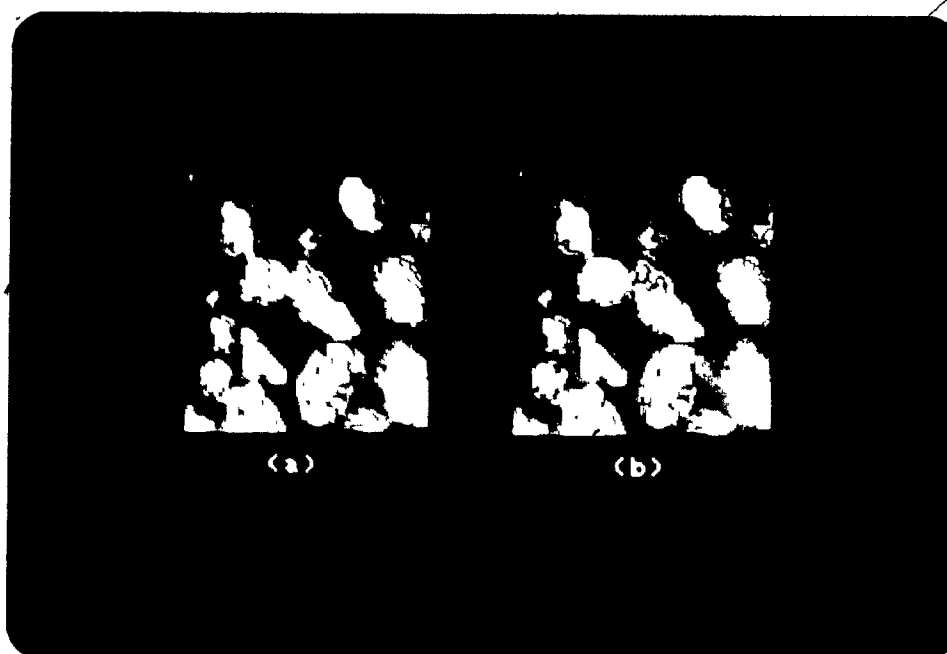


Figure (7.35): Region maps for region rule priority exps.
 (a) Exp. 62: Priority for region merging rules.
 (b) Exp. 63: Priority for region splitting rules.

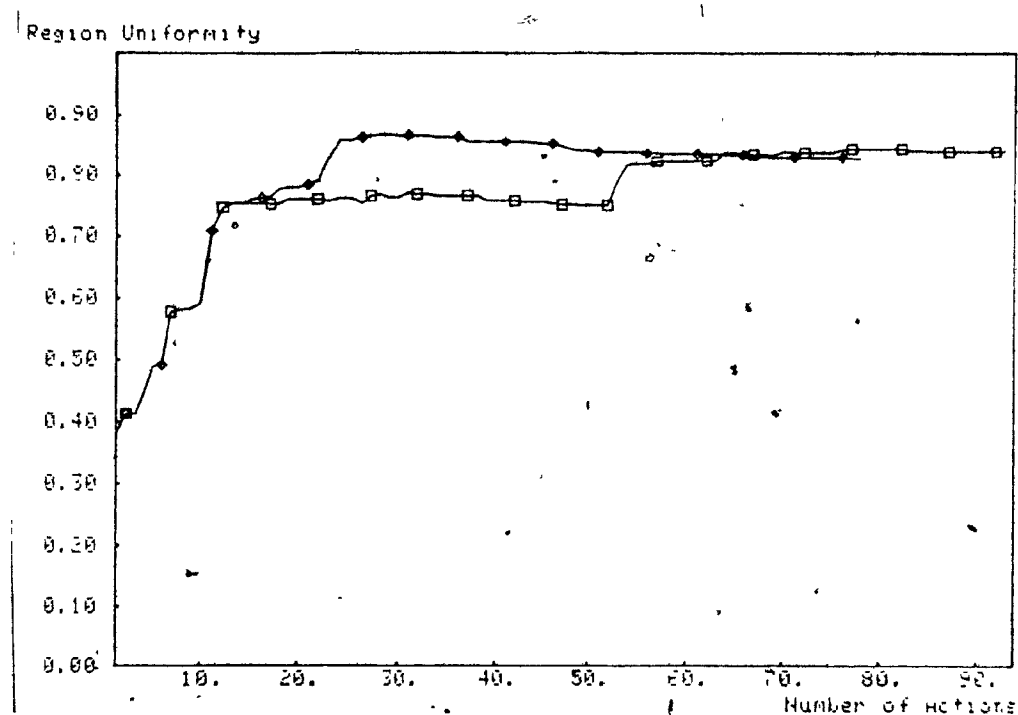


Figure (7.36). Region uniformity for region rule priority.
 Exp. 62 □ Exp. 63 ◇

Exp.	Cycles	Actions	U	C	Under	Over	Total	F
	Initial	0	0.38	0.64	0.00	71.70	50.70	0.00
62	269	93	0.84	0.91	17.54	10.40	14.42	0.99
63	266	78	0.83	0.94	17.23	10.30	14.19	1.01

Table (7.19): Region rule priority experiments.

	LOW	HIGH
NUMBER OF REGIONS	SPLIT	MERGE
REGION UNIFORMITY	SPLIT	MERGE
REGION CONTRAST	MERGE	SPLIT

Table (7.20): The relation between performance measures and region rule priority.

7.3.6. Line Rules Priority Experiments

The initial line map of figure (7.26a) was used in experiment 64 when higher priority was given to rules that delete lines. Rules that add lines were matched first in experiment 65, and the results are compared in table (7.21a) and figures (7.37a) and (7.37b). Priority to line addition produced a lower error and a higher connectivity value. In this case, a large number of initial lines was missing. Therefore, when more lines were deleted before the rules that extend lines could be matched, valuable information was lost. For example, the deleted lines could have been used to guide the extension of other lines. The boundary in the top left corner of figure (7.37a) was missed because a short line in the initial map of figure (7.26a) had been deleted. On the

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
Initial	0	0	0.73	0.30	30.44	61.70	48.65	0.00
64	270	137	0.68	0.82	16.86	31.40	25.20	1.35
65	441	189	0.70	0.94	6.68	28.00	20.35	1.00

(a)

Exp.	Cycles	Actions	H	T	Excess	Missing	Total	F
Initial	1	0	0.54	0.30	55.19	43.50	49.69	0.00
66	577	218	0.72	0.81	16.62	32.80	26.00	1.11
67	643	359	0.59	0.91	44.33	31.80	38.58	0.46

(b)

Table (7.21): Line rule priority experiments.

(a) Initial number of lines = 94

(b) Initial Number of Lines = 178

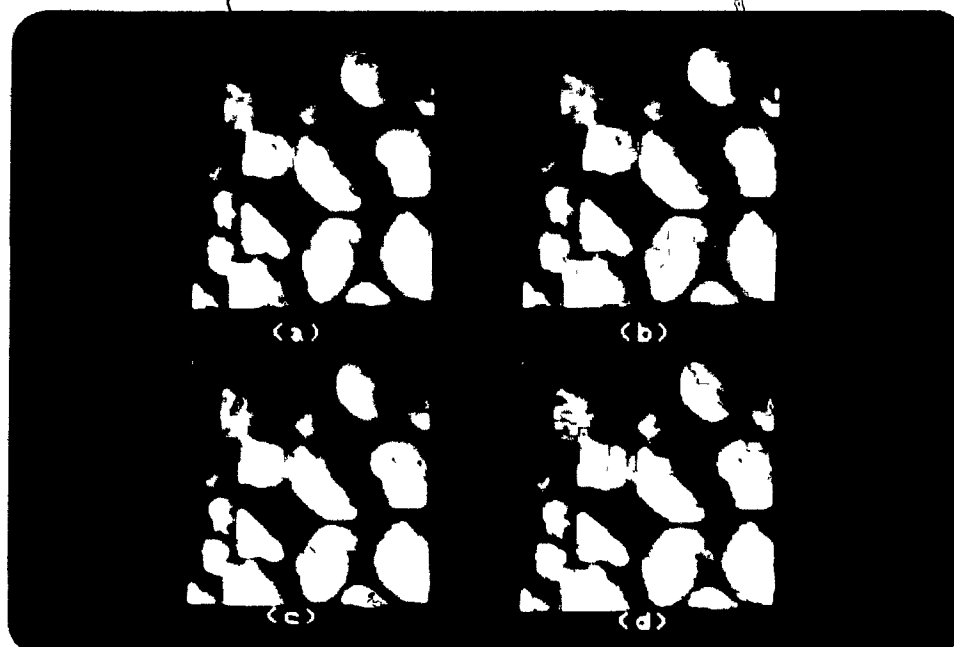


Figure (7.37): Line maps for line rule priority.

(a) Exp. 64 : Deletion priority with low initial lines.

(b) Exp. 65 : Addition priority with low initial lines.

(c) Exp. 66 : Deletion priority with high initial lines.

(d) Exp. 67 : Addition priority with high initial lines.

other hand, in figure (7.37b), when other lines were extended towards that short segment, a perfect representation of the boundary was retained.

Experiments 66 and 67 tested the priority for deletion and addition on the initial line map of figure (7.27a). The resulting measures are shown in table (7.21b) and the output maps are displayed in figures (7.37c) and (7.37d). Because there was a larger number of lines in this initial map, priority to line deletion proved to be superior. This is mainly because it produced a much lower additional lines error. When rules that add lines were given higher priority, they extended and joined lines that should have been deleted in the first place. Therefore, the final output contained a larger number of additional lines, as shown in figure (7.37d). The same conclusion can be derived from the curves in figure (7.38) which show the total error measures for experiments 66 and 67. Priority to line deletion (in experiment 66) resulted in a rapid reduction in the error measure mostly due to the removal of excess lines. This was followed by a slight and steady increase when attempts were made to add and join lines. The curve for experiment 67 showed a much faster increase in the error that corresponded to line additions. When the deleting rules were finally given a chance, they reduced the error substantially, but not to the level achieved by the other curve.

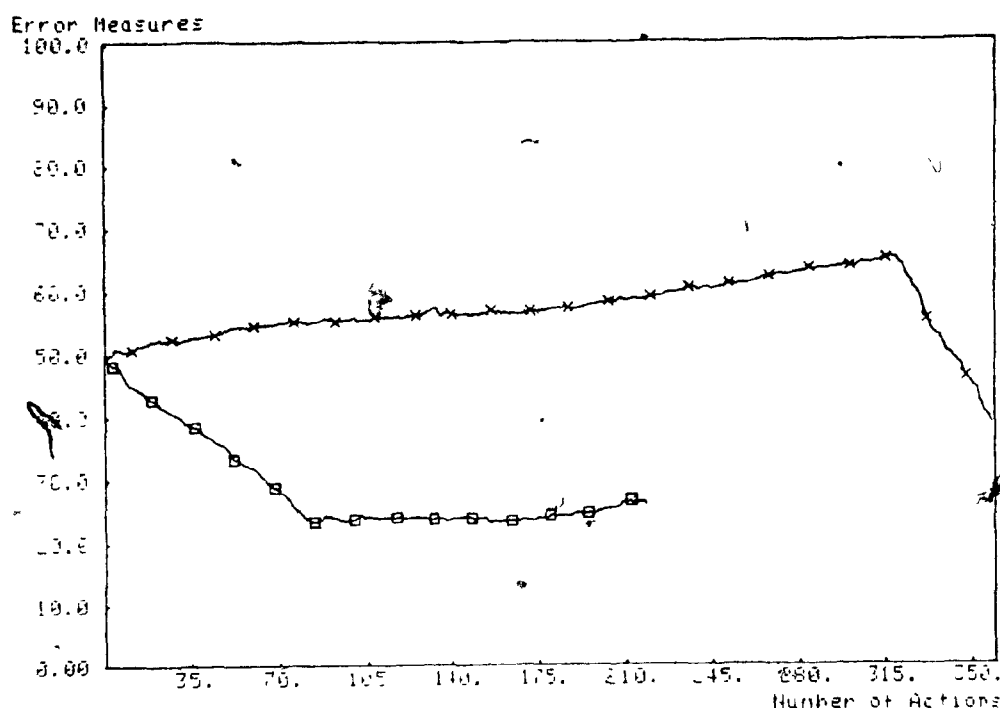


Figure (7.38): Error measures for line rule priority
Exp. 66 □ Exp. 67 x

To summarize, line deletion is a priority if the number of regions is high and if their average contrast is low, and vice-versa. Connectivity improves more when line addition is dominant as apparent from the results of both sets of experiments in table (7.21). Therefore, priority to line addition is preferable when connectivity is low. These constraints are summarized in table (7.22).

	LOW	HIGH
NUMBER OF LINES	ADD	DELETE
LINE CONTRAST	DELETE	ADD
LINE CONNECTIVITY	ADD	DELETE

Table (7.22): The relation between performance measures and line rule priority.

The experiments described above determined for each strategy element the state that produced the best results for different ranges of performance measures. Each of the above sections concluded with a table that summarized the relation between performance measures and one of the strategy elements. The results were coded into the constraint tables described in chapter 6 for dynamic strategy determination. It should be noted that the above constraints were derived by analysing trends in the performance data. These trends were then transformed into the (+/-) codes in table (6.2). In addition, only a subset of the performance measures affected each strategy element. The relative importance of each performance measure in determining a strategy element was quantized into one of the five categories of table (6.4). The results were then stored in table (6.3).

The previous sections described the results of learning experiments for the knowledge rules and the control strategies. The remaining sections use these results to report on the system performance in analysing images of natural scenes.

7.4. FOCUS OF ATTENTION RESULTS

In addition to selecting the knowledge rules and determining the constraints that govern dynamic strategy setting, a third aspect of the system that needed to be tested was the generation of focus of attention areas and their role in control. The rules that generate and maintain

areas of attention were described in section 5.3.3. These rules were used in the analysis in conjunction with the optimal set of region and line rules selected in section 7.2 and the dynamic control strategy. Therefore, a region partition, a line partition, and a set of focus of attention areas were produced at the end of processing. In figure (2.1), the region map was designated to represent the final segmentation output. In addition, the final line map and the focus of attention areas contained supplementary information that may be used in subsequent higher level analysis. This section presents an example of the final segmentation output, as well as the final line map, and the areas generated by the system. The performance measures of these areas will be examined to illustrate how they differed with area type.

The color image of figure (3.1) represents a picture of a typical outdoor scene. The segmentation result is represented by the region map in figure (7.39). The final configuration of lines are shown in figure (7.40). The smooth areas that were computed based on this segmentation are displayed in figure (7.41). They are seen to correspond to the sky, the grassy areas, and the wall. The description of the wall area and its performance vector according to equation (6.22) were given by:

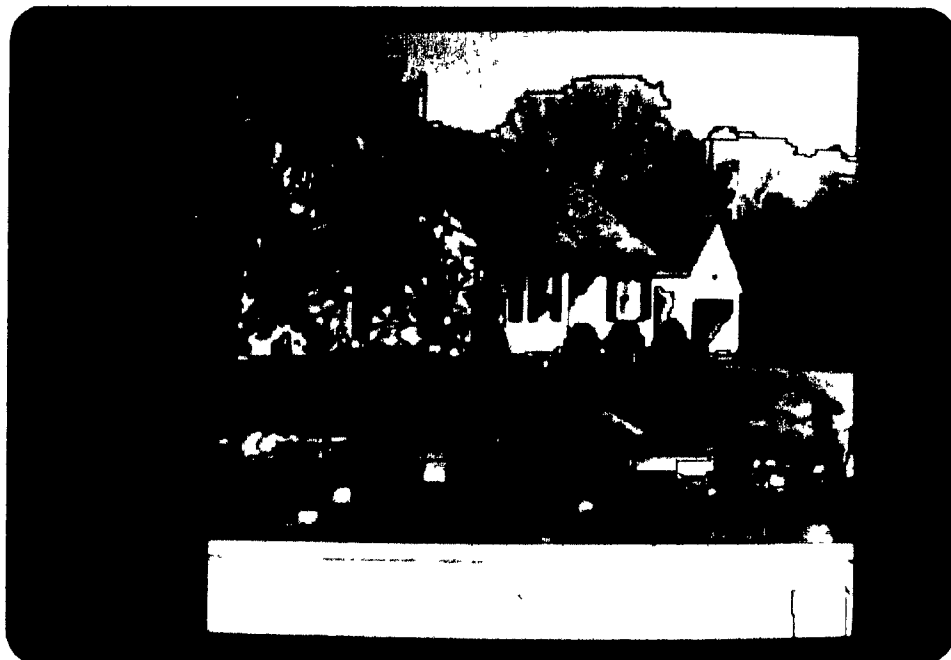


Figure (7 39) The segmentation result (region map)
for the scene in figure (3 1)



Figure (7 40) The final line partition for the scene
in figure (3 1)

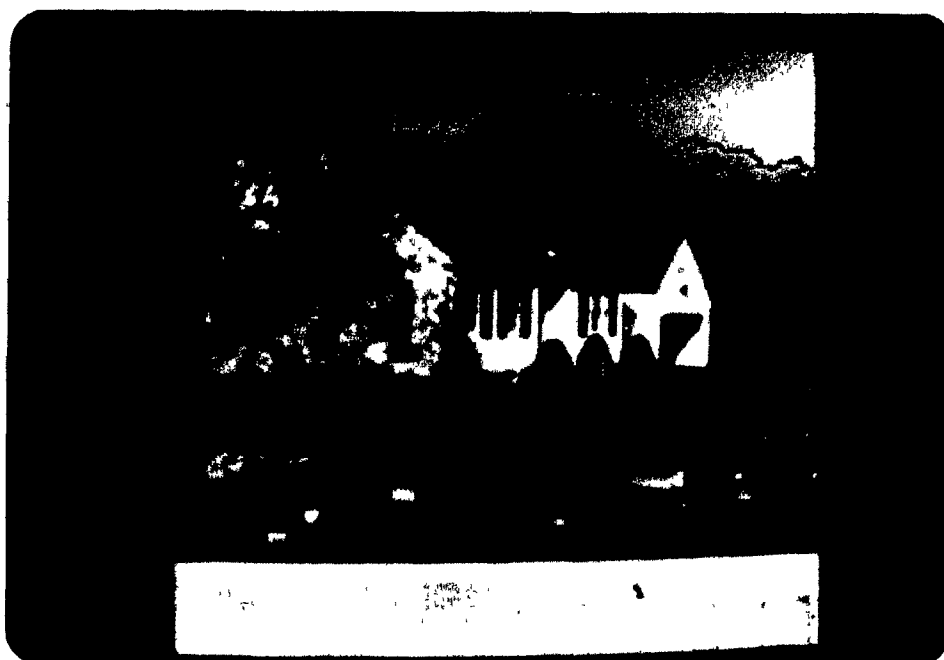


Figure (7 41) : The smooth areas detected in the scene

Information on area 5

Area Type	SMOOTH		
Bounding Rectangle	(2, 2) and (255, 42)		
Area Size	10161		
Total Line Lengths	319		
Number of Regions	3		
Number of Lines	5		
Current Region	3		
Current Line	01		
	Red	Green	Blue
Area Variances	5 68	14 31	9 73
Region Uniformity	0. 97	0 97	0 96
Region Contrast	0 30	0 64	0 65
Line Contrast	0 19	0 25	0 23
Line Connectivity	1. 00		

It is seen that the area is characterized by a high uniformity of the three color features, which is also evident from the low value of the variances. The normalized number of regions is low, since the area is composed of a small

number of large regions. There is a total of five lines in this smooth area. The contrast between these lines is low, and so is the contrast between the three regions that constitute the area.

The strategy vector of equation (6.48) varies during the processing as a function of the performance measures of each area. The elements of strategy were computed using the dynamic selection process defined in section 6.4. The strategy vector that corresponds to the final measurements of area 5 is given by:

Strategy vector for area 5:

```

-----
Region Path Strategy : Region with Smallest Size
Line Path Strategy  : Longest Line
Updating strategy    : Successive
Process Priority     : Lines
Region Rule Priority : Merge
Line Rule Priority   : Delete
  
```

Three textured areas were detected in the scene and are shown in figure (7.42). The performance vector for the large tree is as follows:

Information on area 3

```

-----
Area Type ,           : TEXTURE
Bounding Rectangle : ( 1,128) and ( 94,256)
Area Size          : 8372
Total Line Lengths : 1959
Number of Regions  : 74
Number of Lines    : 178
Current Region     : 50
Current Line       : 210

Area Variances      : Red      Green    Blue
Region Uniformity   : 0.52     0.50     0.56
Region Contrast     : 0.34     0.44     0.49
Line Contrast       : 0.24     0.28     0.34
Line Connectivity   : 0.99
  
```

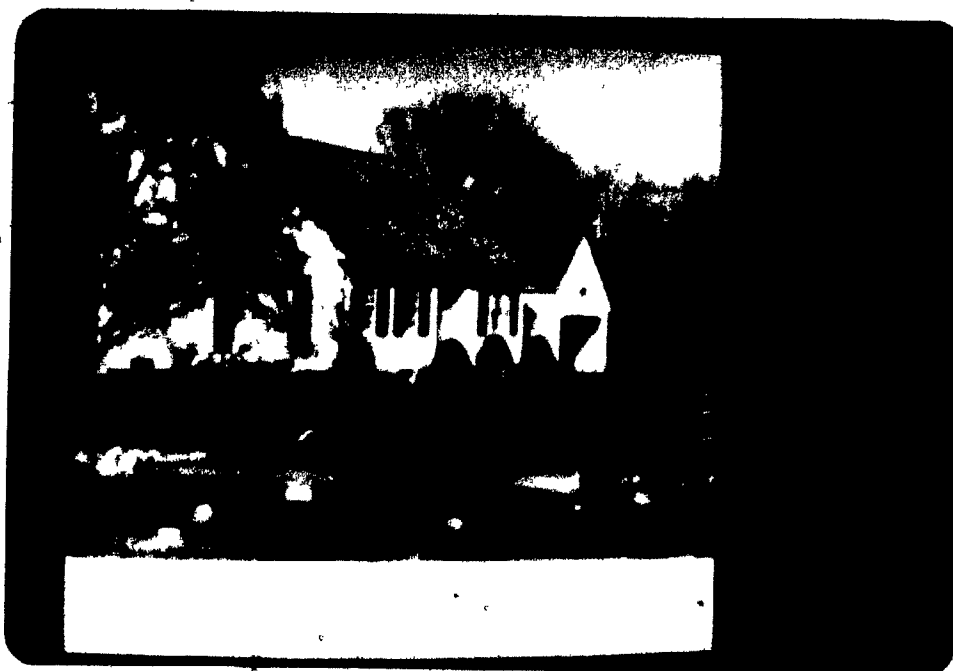



Figure (7.42) : The textured areas in the scene.

The variance of the area is seen to be high. The uniformity within each region is lower than for smooth areas, but the number of regions is much greater (which also reflects the smaller size of these regions). The number of lines is also high, and their connectivity is large. Region and line contrast are seen to exceed the values for smooth areas. The strategy vector at the end of processing corresponds to the final measurements of area 3, and is given by:

Strategy vector for area 3:

Region Path Strategy : Region with Largest Adjacency
 Line Path Strategy : Weakest Line
 Updating Strategy : Single
 Process Priority : Regions
 Region Rule Priority : Split
 Line Rule Priority : Delete

An example of a bounded area is shown in figure (7.43). By examining the line map in figure (7.40), the roof area is seen to be enclosed by a number of lines. By analysing this area separately, it was possible to distinguish the regions that form the roof from the similar regions that belong to the tree at its upper right corner. The performance measurements for the roof area are given by.

Information on area 8

```

-----
Area Type           : BOUNDED
Bounding Rectangle  : ( 79,153) and (178,200)
Area Size           : 2955
Total Line Lengths  : 392
Number of Regions   : 3
Number of Lines     : 20
Current Region      : 10
Current Line        : 4
  
```

	Red	Green	Blue
Area Variances	26.01	13.20	38.54
Region Uniformity	0.86	0.88	0.87
Region Contrast	0.31	0.26	0.32
Line Contrast	0.25	0.24	0.29
Line Connectivity	0.88		

The diversity in the values of the parameters for the given areas underlines their effectiveness in dynamically setting the processing strategy.

7.5. SYSTEM PERFORMANCE

In order to demonstrate the advantages of using the rule-based system, its results are compared in this section to those of earlier approaches to segmentation. The segmentation outputs of the rule-based system were computed using the optimal set of rules and dynamic strategy.



Figure (7 43): Example of a bounded area.

selection Two algorithms were chosen Ohlander's histogram splitting method [Ohlander, 1975] and the split-and-merge algorithm [Tanimoto and Pavlidis, 1975] (*). The first method was chosen to represent the class of region splitting algorithms that are based on histogram analysis The second method was selected because it employs both region merging and splitting, albeit in a very restricted data structure Whereas cluster analysis in feature space is the basis for the first method, the second method uses a structural approach based on similarity grouping

(*) The author wishes to acknowledge the efforts of Mr W. Hong in implementing the second algorithm

A number of parameters are involved in the analysis using both methods. For histogram segmentation, a fixed set of numerical criteria are used to define the peaks within the histogram. The split-and-merge algorithm requires the instantiation of two variables. The starting level of the "initial cutset" refers to the resolution of the square grid that forms the initial region partition. The second parameter is a tolerance level that defines the allowable range of grey level features within one region.

The region partitions that resulted from applying the two methods to the scene of figure (3.1) are shown in figure (7.44). These are compared to the rule-based segmentation output obtained for the same image in figure (7.39). Qualitatively, the latter is seen to exhibit a superior partition that corresponds more closely to the objects in the scene. In addition, a quantitative analysis of the results using the segmentation error measures of equations (3.26) and (3.28) and the reference segmentation of figure (3.7) are summarized in table (7.23).

Method	Outdoor Scene		Blocks	
	Under Merging	Over Merging	Under Merging	Over Merging
Rule-based	18.50	11.16	4.16	3.75
Histogram, splitting	17.77	36.43	1.53	37.37
Split-and-merge	55.16	12.74	40.96	2.54

Table (7.23): Comparison of error measures for three methods.



(a)



(b)

Figure (7.44): Partitions of the OUTDOOR SCENE
(a) using Ohander's histogram splitting
(b) using the split-and-merge algorithm.

Table (7.23) also includes the error measures for the blocks image of figure (7.45a) whose output region partitions are shown in figures (3.45b) to (3.45d). The histogram splitting method is seen to produce high over-merging errors that correspond to the large regions of the image which are not segmented. This is because although these regions satisfy other criteria for splitting (such as high gradient or variances), their histograms fail to meet the necessary conditions for peak selection defined in [Ohlander, 1975]. The method is clearly more suitable for extracting contrasting objects from their background than it is for general purpose segmentation.

On the other hand, the split-and-merge algorithm produced low over-merging errors, but at the expense of high under-merging components. For example, in order to detect the top face of the upper block in figure (7.45d), the tolerance level within regions had to be lowered. This produced too much detail in the foreground. Whereas a tolerance level of 8 was used for the outdoor scene segmentation, 1 was the value used for the blocks image. This parameter dependency became evident when other images were segmented and the values used to obtain best results varied widely. The second parameter for the split-and-merge algorithm was the starting level within the pyramid data structure. The best partitions were obtained when processing started at the highest resolution level, so that no splitting took place. In addition to being more time consuming, this

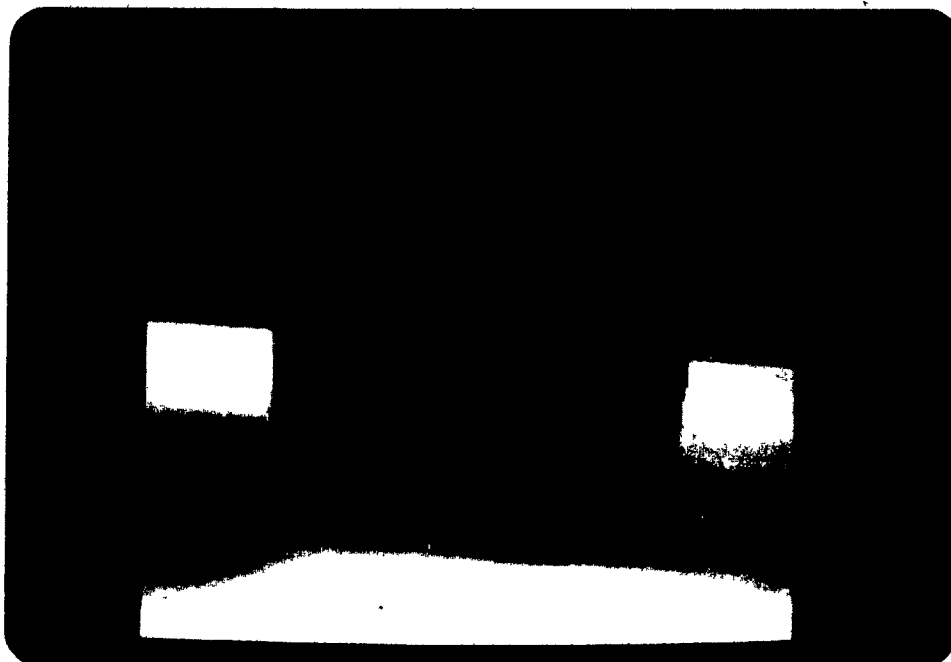


(a)

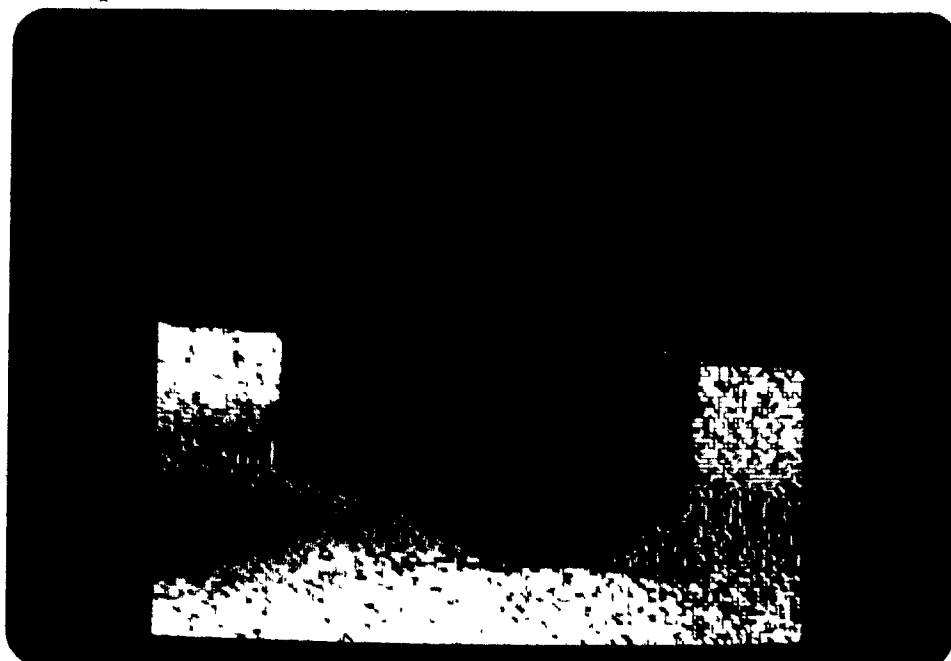


(b)

Figure (7.45). Partitions of the BLOCKS image.
(a) Original image
(b) Rule-based partition.



(c)



(d)

Figure (7 45) contd. Partitions of the BLOCKS image
(c) Ohlander's histogram splitting.
(d) The split-and-merge algorithm

defied the purpose of the algorithm because it was reduced to a simple region grower

The split-and-merge algorithm was consistently faster than the others with an average cpu time of 370 seconds. This is compared to 840 seconds for histogram segmentation and 1824 seconds for rule-based segmentation (an approximate ratio of 1 : 2.25 : 5). The rigidity of the data structure, the ease of applying the merging and splitting criteria, and the simplicity of the two basic actions were all factors that contributed to the speed of the split-and-merge process. As the data structure became more flexible, more time was needed to compute and analyse the histograms of arbitrarily shaped regions and then split them accordingly. The rule-based system combined histogram analysis with other region and line analysis techniques in a formulation that maintained flexibility of data and control. However, a price was paid in terms of the increased computation time required for its test and modify cycles.

Figures (7.46) to (7.48) compare the results of the three methods for the HOUSE, GIRL, and CAR color images. Again, even when the parameters were adjusted for best output, the split-and-merge algorithm tended to over-segment the image. Histogram splitting separated the more prominent objects from the background without completely segmenting the image. The output of the rule-based system conformed more with human expectations of low level segmentation with almost no over-merging errors and few undermerged regions.



(a)



(b)

Figure (7 46) Partitions of the HOUSE image
(a) Original image
(b) Rule-based partition



(c)



(d)

Figure (7 46) contd Partitions of the HOUSE image
(c) Ohlander's histogram splitting
(d) The split-and-merge algorithm



(a)



(b)

Figure (7 47) Partitions of the GIRL image
(a) Original image
(b) Rule-based partition

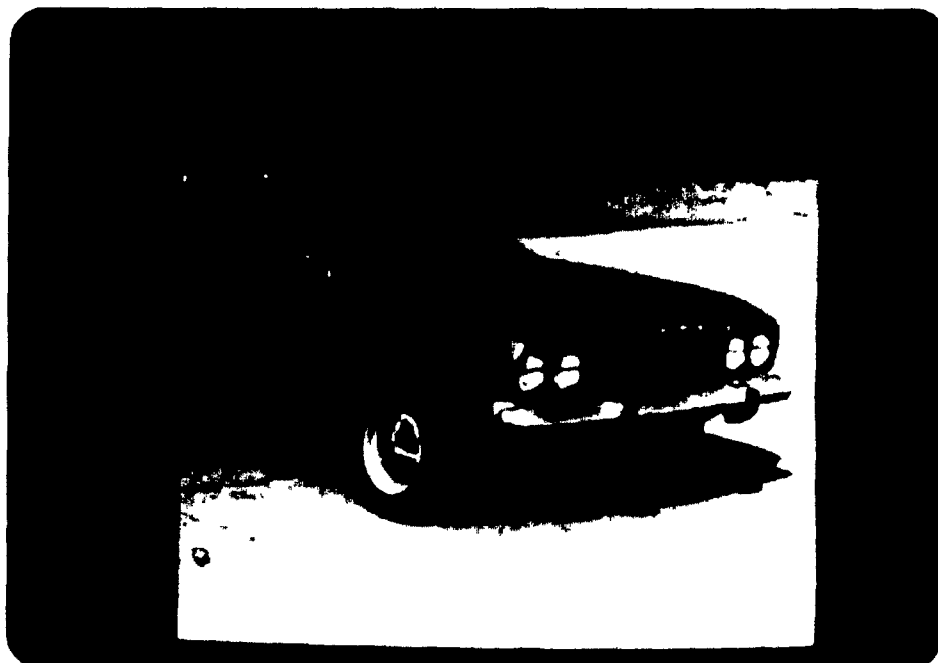


(c)

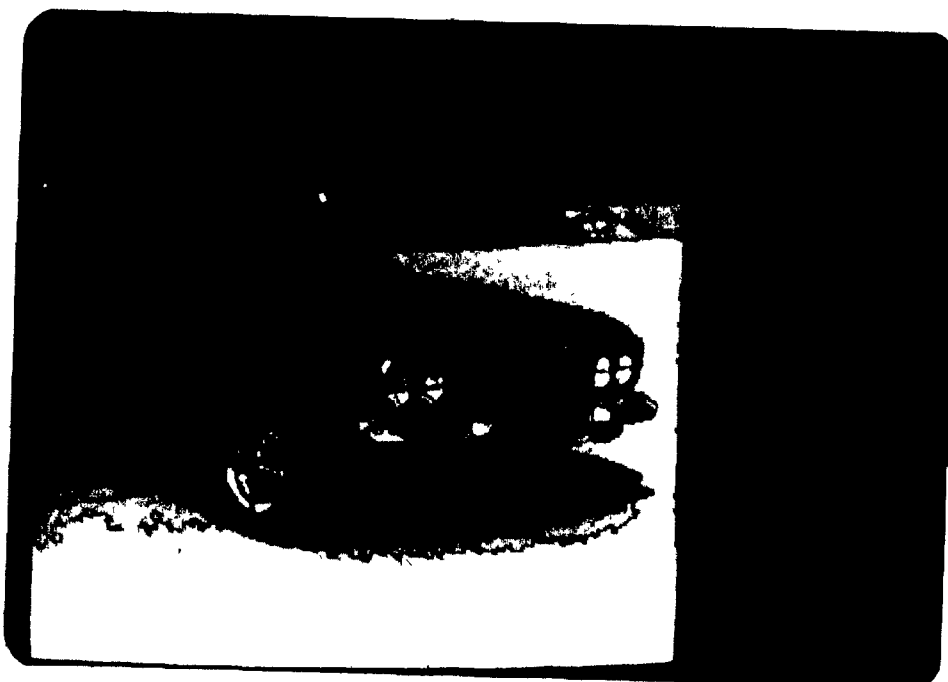


(d)

Figure (7.47) contd Partitions of the GIRL image
(c) Ohlander's histogram splitting
(d) The split-and-merge algorithm

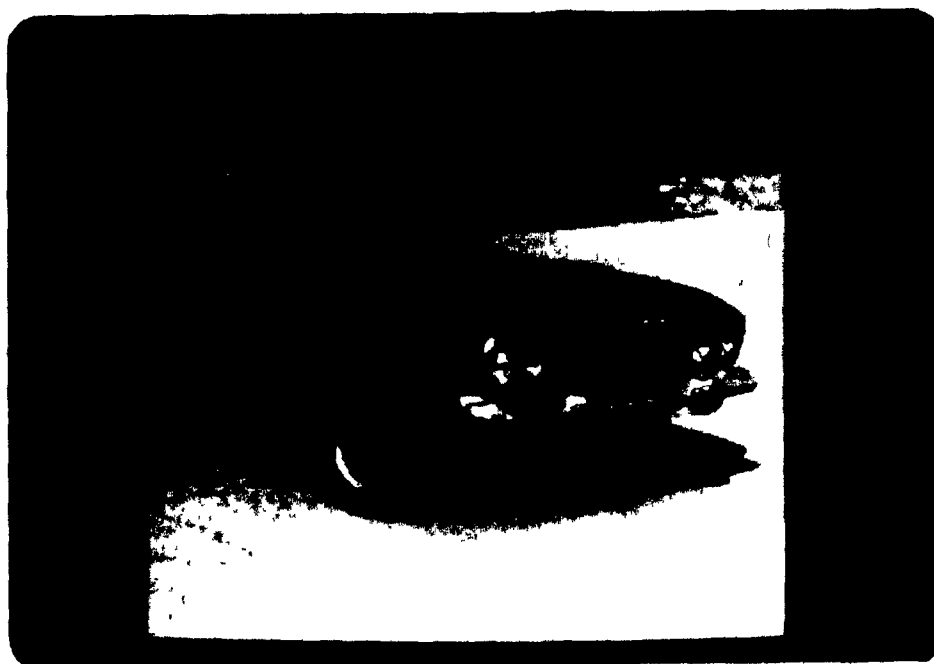


(a)

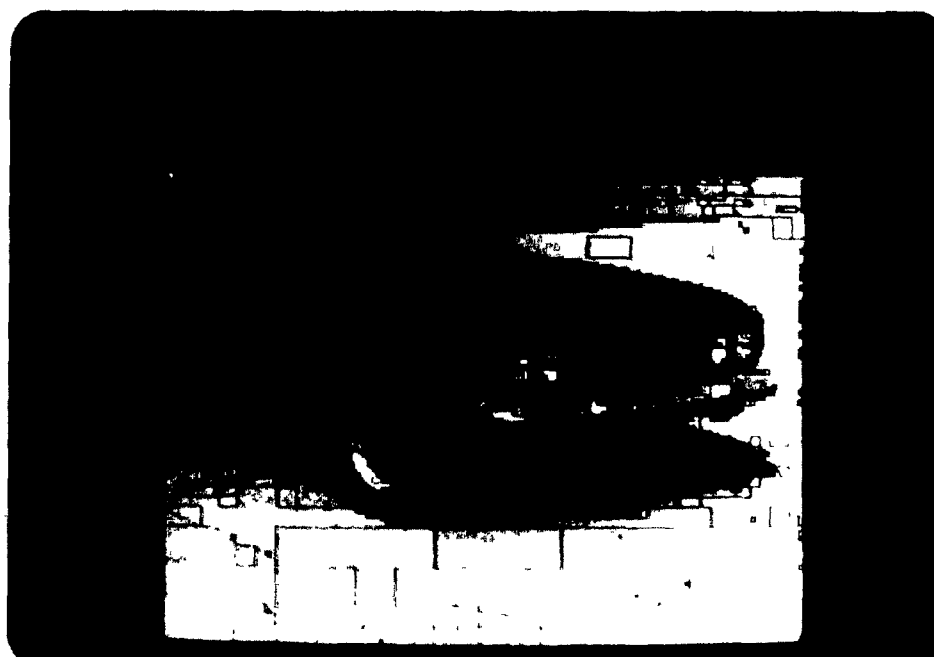


(b)

Figure (7 48) Partitions of the CAR image
(a) Original image
(b) Rule-based partition



(c)



(d)

Figure (7 48) contd. Partitions of the CAR image
(c) Ohlander's histogram splitting
(d) The split-and-merge algorithm

The region uniformity and region contrast performance measurements for the final region maps of several images are summarized in table (7 24) for the three methods. Once again, the results agreed with qualitative evaluation and the segmentation error measures. The split-and-merge algorithm produced regions with high uniformity for the three color features, but with low contrast values to their adjacent neighbors. This is a clear indication of over-segmentation. On the other hand, Ohlander's histogram splitting resulted in highly contrasting regions that are less uniform. Therefore, further segmentation was needed to improve uniformity.

Rule-based segmentation improved the two measures simultaneously, as is evident from the two-dimensional histogram of figure (7 49). The average uniformity of the color features is plotted against the average contrast for the three processes over the same five images. Values for the split-and-merge algorithm cluster near the high uniformity-low contrast end of the graph, while those for histogram analysis fall at the low uniformity-high contrast end. Points that belong to the rule-based system are seen to group at high values for both measures.

The results presented in this section show a significant improvement in the output segmentations that resulted from using the rule-based system over those that were obtained from simpler region analysis methods. This was detectable in the quality of the final partitions, as well as in the segmentation error and the performance measures. The

		Rule-based		Split-and-merge		Histogram	
		U	C	U	C	U	C

OUTDOOR SCENE	R	0 77	0 49	0 77	0 32	0 59	0 43
	G	0 80	0 60	0 81	0 38	0 63	0 50
	B	0 80	0 66	0 77	0 42	0 55	0 56
Average		0 79	0 58	0 79	0 37	0 59	0 50
CPU Time		2317		536		1254	

BLOCKS	R	0 98	0 64	0 99	0 45	0 69	0 78
	G	0 95	0 65	0 99	0 45	0 59	0 77
	B	0 95	0 63	0 99	0 36	0 59	0 80
Average		0 96	0 64	0 99	0 42	0 63	0 78
CPU Time		1327		219		529	

HOUSE	R	0 87	0 66	0 92	0 40	0 72	0 63
	G	0 85	0 64	0 93	0 37	0 68	0 61
	B	0 87	0 65	0 95	0 36	0 72	0 61
Average		0 87	0 65	0 94	0 38	0 70	0 62
CPU Time		2083		427		898	

GIRL	R	0 88	0 52	0 88	0 32	0 75	0 59
	G	0 86	0 57	0 86	0 34	0 71	0 66
	B	0 85	0 60	0 86	0 34	0 53	0 69
Average		0 86	0 57	0 87	0 34	0 67	0 65
CPU Time		1657		271		609	

CAR		0 85	0 56	0 91	0 40	0 44	0 75
		0 84	0 54	0 88	0 39	0 41	0 72
		0 85	0 55	0 89	0 39	0 44	0 74
Average		0 85	0 55	0 90	0 40	0 43	0 74
CPU Time		1738		395		912	

Table (7 24) A comparison of the performance measurements of three approaches

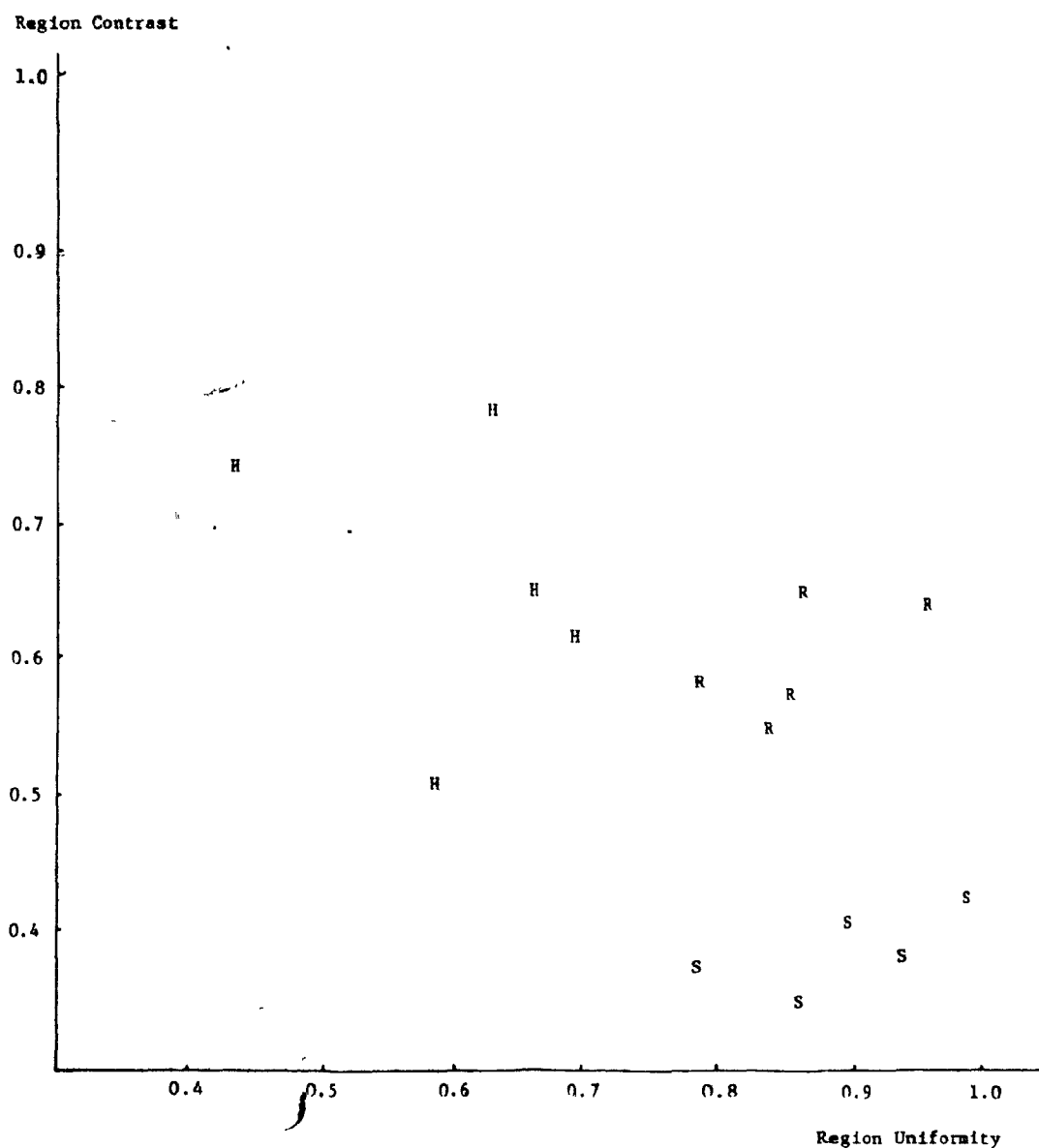


Figure (7. 49): A two-dimensional histogram of performance measures for the three methods.

S : Split-and-merge.

H : Histogram splitting.

R : Rule-based segmentation.

complexity of the system, the added knowledge, and the intelligent control structure produced substantial gains, but not without the inevitable cost of increased computational time.

7.6. CONCLUSIONS AND FUTURE WORK

The system presented in this thesis provides a new view of and a more general solution to the image segmentation problem. Earlier approaches have mostly created a scene partition by designing procedures that apply one or two heuristics in a uniform way and in a prespecified order to the image data. Because of their rigid structure and the limited knowledge they used, the outputs were not sufficiently developed for the interpretation systems that followed. The rule-based segmentation system furnishes a needed buffer between the procedural operators that sample the image data in order to detect and group features on the one hand, and the high level systems that require the best possible partition in order to apply their domain specific knowledge.

Although it is an expert system, the knowledge used by the rule-based segmenter is not semantically dependent on the contents of the image under analysis, as is the case with the expert interpretation systems. Rather, the rules embody general purpose knowledge about the structural and statistical features of images. For example, Gestalt rules of grouping that are based on concepts like similarity, smoothness, and continuity are coded in a modular symbolic form. Expressing the low level knowledge explicitly makes it compatible with knowledge-based interpretation systems. This provides a common ground for establishing feedback paths through which more low level analysis can be requested.

The suitability and usefulness of the rule-based approach to solving the computer vision problem have been discussed in chapter 2. The data structure presented in chapter 3 reveals that both uniformity analysis in terms of regions, and discontinuity analysis in terms of lines are employed. Previous image understanding systems have depended mainly on one or the other, or when both were used, separate paths were taken and the results later combined. In contrast, this system allows region and line analysis to interact throughout the processing. The exchange of information through the simultaneous application of the two processes was shown in this chapter to produce results that are superior to those of each process alone. Another aspect of the data structure was the introduction of the focus of attention areas as tools for controlling the flow of processing in the image. They were also shown to play an important role in detecting the occurrence of texture. Chapter 4 described how the processing modules modified the data structure to transform the input image measurements into final output partitions. A hierarchy of independent actions provided the means by which the model rules attempted to conform the measured and computed data with the criteria specified by the rule conditions.

In chapter 5, a scheme for the symbolic coding of the rules was presented. The conditions and actions were shown to constitute a segmentation language for expressing the rules. This was followed by a discussion of the rules for

analyzing regions, lines, and areas. Next, two types of control rules were introduced. These were the focus of attention rules for specifying the processing path within the image, and the meta-rules for determining the order in which the knowledge rules were applied. Different processing strategies were employed by selecting the appropriate set of control rules. In chapter 6, the components of the rule-based strategy for processing images were defined, and each was assigned specific states. A dynamic strategy determination process that used a fuzzy decision-maker was then introduced. The latter was based on using performance measurements from the image to select the best local strategy. The control strategy adapted to the properties of the individual areas in the image. Moreover, the properties of the areas varied with time. Therefore, the control strategy was automatically adjusted accordingly, so that the most suitable strategy was applied at any point in time.

The success of the dynamic strategy mechanism depends on the method by which performance is measured. In fact, two types of measures were developed during the course of this research. The segmentation error measures described in chapter 3 compared the image partition with one that is manually generated. Although it was dynamically evaluated, it could only serve as a means for training the model by testing the rules. On line measures that did not require a reference segmentation were then introduced in chapter 6 to guide the strategy selection process. They incrementally

evaluated the state of the segmentation as the processing progressed. This was accomplished by updating global measures, such as the average uniformity of the regions, and the average contrast across lines

Both sets of measures were used in the experiments presented in this chapter. The performance curves and output partitions obtained with different combinations of knowledge rules were compared and analyzed. In addition, each rule was monitored by separately accumulating its contributions to the measures, as well as the frequency at which it was used. These results were used to rank the rules and select an optimal subset that is representative of all the model criteria. A second class of experiments was devoted to examining the effects of the various components of control strategy on the performance measures. By analyzing the behavior of the measures, the constraints that govern their relations to the different strategy states were determined. These constraints define the dynamic decision-making process that selects the appropriate strategies. The latter was then used together with the optimal set of knowledge rules to analyze other images of natural scenes, and the results were compared to other segmentation algorithms. The expanded knowledge base and flexible control structure were shown to contribute a substantial improvement to the resulting output partitions.

Based on the experiences gained in designing, implementing, and experimenting with the expert rule-based segmenter, the following suggestions are made for future paths to explore:

(1) As with all rule-based systems, performance can only be as good as the knowledge used in the rules. The process of finding additional sources of information is a continuing one. More effort should be directed towards the discovery and testing of general purpose segmentation criteria similar to those presented in this thesis. The separation of the knowledge and its generalized symbolic coding permit simple addition of rules whenever new criteria are discovered. The performance measures and the segmentation error measures can then be used to evaluate the new rules in the same manner described earlier in this chapter.

(2) The knowledge rules presented in chapter 5 can be described as "exact inferences" because each condition is a binary logic operation with an outcome that is either true or false. For example, testing whether the value of a feature falls within a finite range. One disadvantage of this knowledge representation scheme is that it is more vulnerable to quantization errors. This is especially true when the criteria to be coded are represented in terms of linguistic, rather than mathematical form. In chapter 6 we presented a fuzzy decision-maker which implemented the constraints that govern the relationship between the performance measurements

and various control strategy components. The same method could be extended to the knowledge rules by defining the conditions as fuzzy variables. Thus, rule matching becomes a fuzzy process that determines how well each condition is satisfied. The rule whose conditions are best satisfied is then selected to execute its actions. For example, the fact that the value of a feature is low can be represented by a fuzzy set that assigns confidence levels to the possible range of feature values. The whole condition is then given the confidence level that corresponds to the particular value being tested. A detailed discussion of inference procedure with uncertainty is given [Ishizuka et al, 1981]. Once the set of knowledge rules has been selected, the criteria can be implemented as fuzzy decision processes. A price is paid in terms of the extra computations needed because the entire rule set has to be evaluated every matching cycle. It remains to be seen whether a worthwhile improvement is achieved by using this formulation.

(3) In addition to brightness measurements (grey level and color), other features of regions and lines can be used in the analysis. In particular, surface orientation is gaining a lot of attention as an important attribute for partitioning scenes [Kanade, 1978, Barrow and Tenenbaum, 1978]. Abrupt changes in orientation will occur at the boundary between two surfaces, while uniform values will exist as orientations change smoothly within a surface. The key issue is how to compute this feature from the measurements in the image.

[Horn, 1977, Woodham, 1977, Pentland, 1982] Whether computed or measured at each pixel in the image, it would be interesting to determine whether surface orientations satisfy the same inferences about uniformity and discontinuity that apply to image intensities. Efforts along these lines are currently under way (*). Although the experiments described above used only brightness as the basic feature, the analysis and implementation have taken into account the adoption of such new features.

(4) Within the context of image understanding, the rule-based segmentation system must be interfaced to an image interpretation system. The function of the latter is to assign real world interpretations to the partitions in the image [Levine and Shaheen, 1981]. In addition to the bottom up flow of data and control that occurs when the output segmentation is delivered to the high level stage, a more flexible interface is desirable. For example, the high level stage may decide to resegment an area of the image if its current partitioning does not conform to the semantic content that has been assigned to its surroundings. This top down flow of control may also be accompanied by some guiding information. The semantic interpreter may signal that a certain area needs more region merging or splitting. The rule-based segmenter will then raise the priority of the

(*) F. Ferrie, a graduate student in the Electrical Engineering Dept., McGill University, is currently investigating these ideas.

corresponding rules to comply with such requests. In fact, the knowledge modularity and dynamic strategy of the segmentation system are ideal for this type of interaction. Feedback can be introduced by allowing the high level stage to influence dynamic strategy setting. This can be accomplished by adding constraints similar to those obtained from the low level performance measures. Therefore, when a focus of attention area is passed back for further analysis, the rule priorities and data selection schemes will be influenced by the information from the interpreter, as well as the performance measurements in the area.

In conclusion, the rule-based segmentation system presented in this thesis addressed a principle aspect in the problem of creating an artificial image understanding scheme. The basic structure separated the knowledge and the control strategy from the processes that implement them. This allowed for the coding of various segmentation criteria and processing strategies in a modular form that was easier to test and evaluate. The results were used to create a model for general purpose image segmentation that is controlled by a dynamic data driven mechanism. It is hoped that the system will influence the design of other aspects of the image understanding hierarchy. Furthermore, other stages in the analysis can contribute new sources of information in the form of new segmentation rules and additional constraints to govern the dynamic strategy determination process.

REFERENCES

- Abdou, I E , Pratt, W K , Quantitative Design and Evaluation of Enhancement/Thresholding Edge Detectors, Proc IEEE, Vol 67, No 5, pp 753-763, May 1979
- Baird, M L , Image Segmentation Techniques for Locating Automotive Parts on Belt Conveyors, Proc 5th IJCAI, pp 694-695, Cambridge, Massachusetts, August 1977
- Barrow, H G , Tenenbaum, J M , Recovering Intrinsic Scene Characteristics from Images, in "Computer Vision Systems", A R Hanson and E M Riseman (eds), Academic Press, New York, pp 3-26, 1978
- Bellman R , Zadeh, L A. , Decision Making in a Fuzzy Environment, Management Sci , Vol 17, pp 144-164, 1970
- Brice, C , Fennema, C , Scene Analysis Using Regions, Artificial Intelligence, Vol 1, pp 205-226, 1970
- Brooks, R.A , Greiner, R. , Binford, T O. , The ACRONYM Model-Based Vision System, Proc 4th IJCAI, pp 105-113, Kyoto, Japan, November 1978
- Bryant, D J , Bouldin, D W. , Evaluation of Edge Operators Using Relative and Absolute Grading, Proc. IEEE Comp Soc Conf on Pattern Recognition and Image Processing, Chicago, IL, pp 138-145, August 1979.
- Buchanan, B.G , Smith, D H. , White, W.C , Gitter, R.J , Feigenbaum, E A. , Lederberg, J , Djerassi, C. , Applications of Artificial Intelligence for Chemical Inference XXIII. Automatic Rule Formation in Mass Spectrometry by Means of the META-DENDRAL Program, Journal of the ACS, Vol 98, pp 61-68, 1976.
- Bullock, B.L. , Unstructured Control and Communication Processes in Real World Scene Analysis, Hughes Research Laboratory, Computer Science Report CS-1, Malibu, CA, August 1977

- Campbell, F W , Robson, J.G. , Application of Fourier Analysis to the Visibility of Gratings, Journal of Physiology, London, Vol 197, pp 551-566, 1968
- Clowes, M. B , On Seeing Things, Artificial Intelligence, Vol. 2, No 1, pp. 79-116, 1971
- Codd, E F. , A Relational Model for Large Shared Data Banks, CACM 13, No. 6, June 1970
- Coleman, G. B , Image Segmentation by Clustering, Technical Report, Image Processing Institute, University of Southern California, USCIP Report 750, Los Angeles, California, July 1977
- Davis, L S , A Survey of Edge Detection Techniques, Computer Graphics and Image Processing, Vol 4, pp 248-270, 1975
- Davis, R , Applications of Meta Level Knowledge to the Construction, Maintenance and Use of Large Knowledge Bases, Memo HPP-76-7, Computer Science Department, Stanford University, Stanford, CA, 1976
- Davis, R., King J J., An Overview of Production Systems, in "Machine Representations of Knowledge, Elcock and Mechie (eds), John Wiley, pp 300-334, 1977
- Davis, R., Meta-rules Reasoning about Control, Artificial Intelligence, Vol. 15, pp 179-222, 1980
- Denofsky, M. E., "How Near is Near?", MIT-AI Memo-334, MIT, AI Lab , February 1976.
- Douglas, R. J., Interpreting Three-Dimensional Scenes A Model Building Approach, Computer Graphics and Image Processing, Vol. 17, No. 2, pp. 91-113, October 1981
- Duda, R. O., Hart, P. E., "Pattern Recognition and Scene Analysis", p. 271, John Wiley and Sons, New York, 1973
- Duda, R. O., Hart, P E., Nilsson, N J , Sutherland, G L , Semantic Network Representations in Rule-Based Inference Systems, in "Pattern-Directed Inference Systems", D Waterman and F. Hayes-Roth (eds), Academic Press, New York, pp. 203-221, 1978.
- Ellis, W. D., "A Source Book of Gestalt Psychology", Kegan Paul, Trench, Trubner and Co., Ltd., London, 1938
- Erman, L. D., Fennel, R. D., Lesser, V R., Reddy, D R., System Organizations for Speech Understanding. Implementation of Network and Multiprocessor Computer Architectures for AI, Proc. 3rd IJCAI, pp 194-199, Stanford, CA, 1973 .

- Falk, G., Computer Interpretation of Imperfect Line-Data as a Three-Dimensional Scene, Stanford AI Memo AIM-132, Stanford University, Stanford, CA, August 1970
- Feigenbaum, E. A., Buchanan, B. G., Lederberg, J., On Generality and Problem Solving. A Case Study Using the DENDRAL Program, Machine Intelligence 6, Edinburgh Univ Press, 1971
- Feigenbaum, E. A., The Art of Artificial Intelligence 1. Themes and Case Studies of Knowledge Engineering, Memo HPP-77-25, Computer Science Department, Stanford University, Stanford, CA, August 1977.
- Fram, J. R., Deutsch, E. S., On the Quantitative Evaluation of Edge Detection Schemes and their Comparison with Human Performance, IEEE Trans on Computers, Vol C-24, No 6, pp 616-628, June 1975
- Freeman, H., On the encoding of arbitrary geometric configurations, IRE trans, EC-10, pp 260-268, 1961
- Gaines, B. R., Foundations of Fuzzy Reasoning, Int J Man-Machine Studies, Vol 8, pp. 623-668, 1976
- Gaines, B. R., Kohout L. J., The Fuzzy Decade: a Bibliography of Fuzzy Systems and Closely Related Topics, Int J Man-Machine Studies, Vol 9, pp. 1-68, 1977.
- Guzman, A., Decomposition of a Visual Scene into 3-D Bodies, AFIPS 1968 Fall Joint Computer Conference, 33, Part 1, pp. 291-304, 1968.
- Hanson, A. R., Riseman, E. M., Nagin, P., Region Growing in Textured Outdoor Scenes, Proceedings of the Third Milwaukee Symposium on Automated Computation and Control, pp. 407-417, 1975.
- Hanson, A. R., Riseman, E. M., Segmenting Natural Scenes, in "Computer Vision Systems", A. Hanson and E. Riseman (eds.), Academic Press, New York, 1978.
- Haralick R. M., Shanmugan, K., Computer Classification of Reservoir Sandstones, IEEE Trans. Geoscience Electronics, GE-11, No. 4, pp 171-177, October 1973
- Horn, B. K., Understanding Image Intensities, Artificial Intelligence, 8, No. 2, pp. 201-231, 1977.
- Horowitz, S. L., Pavlidis, T., Picture Segmentation by a Directed Split-and-Merge Procedure, Proc Second Int. Joint. Conf. on Pattern Recognition, pp. 424-433, 1974.

- Huffman, D A , Logical Analysis of Pictures of Polyhedra, SRI Technical Note No 6, AI group, Stanford Research Institute, Stanford, CA, May 1969
- Ishizuka, M., Fu, K S , Yao, I T P , Inference Procedure with Uncertainty for Problem Reduction Method, TR-EE 81-33, School of Electrical Eng , Purdue University, Indiana, August 1981
- Joshi, A K , Some Extensions of a System for Inferencing on Partial Information, in "Pattern Directed Inference Systems", D Waterman and F Hayes-Roth (eds), Academic Press, pp 241-257, 1978
- Kanade, T , Region Segmentation Signal vs Semantics, Proc 4th IJCPR, pp 95-105, Kyoto, Japan, Nov 7-10, 1978
- Kirsch, R , Computer Determination of the Constituent-Structure of Biological Images, Computers and Biomedical Research, Vol 4, pp 315-328, 1971
- Kitchen, L , Rosenfeld, A , Edge Evaluation Using Local Edge Coherence, IEEE Trans on Systems, Man, and Cybernetics, Vol SMC-11, No 9, pp. 597-605, September 1981
- Koffka, K , "Principles of Gestalt Psychology", Harcourt, Brace & World Inc , New York, 1963
- Kohler, W , "The Task of Gestalt Psychology", Princeton University Press, Princeton, New Jersey, 1969
- Land, E.H , The Retinex Theory of Color Vision, Scientific American, 237, No 6, pp 108-128, December 1977.
- Lesser, V R , Fennell, R.D., Erman, L D., Reddy, D.R., Organization of the HEARSAY-II speech understanding system, IEEE Trans on Acoustics, Speech, and Signal Processing, ASSP-23, No 1, pp 11-23, Feb 1975.
- Lesser, V.R , Erman, L D , A Retrospective View of the HEARSAY-II Architecture, Proc 5th IJCRI, pp. 790-800, Boston, MA, 1977.
- Levine, M.D , Leemet, J., A Method for Non-Purposive Picture Segmentation, Proc. 3rd IJCPR, San Diego, CA, November 1976.
- Levine, M.D , A Knowledge-Based Computer Vision System, in "Computer Vision Systems", A.R Hanson and E.M. Riseman (eds), Academic Press, New York, pp. 335-352, 1978.

- Levine, M D , Region Analysis with a Pyramid Data Structure, in "Structured Computer Vision Machine Perception Through Hierarchical Computation Structures", S Tanimoto and A Klinger (eds), pp 57-100, 1980.
- Levine, M D , Ting, D , Intermediate Level Picture Interpretation Using Complete Two-Dimensional Models, Computer Vision and Image Processing, Vol. 16, No. 3, pp 185-209, July 1981
- Levine, M D , Shaheen, S I , A Modular Computer Vision System for Picture Segmentation and Interpretation, IEEE trans on Pattern Analysis and Machine Intelligence, Vol PAMI-3, No 5, pp 540-556, September 1981
- Levine, M D , Nazif, A , An Experimental Rule-Based System for Testing Low Level Segmentation Strategies, in "Multicomputers and Image Processing Algorithms and Programs", K Preston and L Uhr, (eds), Academic Press, N Y , pp 149-160, 1982a
- Levine, M D , Nazif, A , Performance Measurement and Strategy Evaluation for Rule-Based Image Segmentation, TR82-1, Computer Vision and Graphics Lab , Dept of Elec Eng , McGill University, March 1982b
- Levine, M D , "Vision in Man and Machine", McGraw-Hill, in preparation (1983)
- Marr, D , Early Processing of Visual Information, Phil Trans R Soc London B 275, pp. 483-524, 1976
- Marr, D , Representing Visual Information, in "Computer Vision Systems", A R Hanson and E M Riseman (eds), Academic Press, New York, 1978.
- Marr, D , Hildreth, E , Theory of Edge Detection, Proc. R. Soc London B 207, pp 187-217, 1980.
- Marr, D. , "Vision", Freeman and Co , San Francisco, 1982.
- Matsuyama T , Saburi K , Nagao, M , A Structural Analyzer for Regularly Arranged Textures, Computer Graphics and Image Processing, Vol. 18, No 3, pp 259-278, March 1982.
- McKee, J W , Aggarwal, J.K. , Finding the Edges of the Surfaces of Three-Dimensional Curved Objects by Computer, Pattern Recognition, Vol. 7, pp. 25-52, 1975.
- Moran, J P , The Symbolic Nature of Visual Imagery, Proc. 3rd IJCAI, pp 472-477, Stanford, CA, August 1973.

- Muerle, J.L., Allen, D.C., Experimental Evaluation of Techniques for Automatic Segmentation of Objects in a Complex Scene, in "Pictorial Pattern Recognition", G.C. Cheng et al (eds), pp 3-13, Thompson, Washington, 1968.
- Nakagawa, Y., Rosenfeld, A., Some Experiments in Variable Thresholding, Pattern Recognition, Vol 11, No 3, pp. 191-204, 1979
- Nazif, A., A Survey of Color, Boundary Information, and Texture as Features for Low Level Image Processing, Report No 78-7R, Computer Vision and Graphics Lab, Dept of Electrical Eng, McGill University, Montreal, Quebec, 1978
- Nazif, A., Levine, M.D., A Rule-Based Image Segmentation System, Seventh Conference of the Canadian Society on Man-Computer Studies, Waterloo, Ontario, Canada, May 10-12, 1981
- Nazif, A., Levine, M.D., Setting Strategy in a Rule-Based Vision System, Proceedings of the Fourth International Conference of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Saskatchewan, Canada, May 17-19, 1982
- Newell, A., Simon, H.A., "Human Problem Solving", Prentice Hall, Englewood Cliffs, NJ, 1972
- Newell, A., Production Systems, Models of Control Structures, in "Visual Information Processing", Academic Press, New York, 1973
- Nii, H.P., Feigenbaum, E.A., Rule-Based Understanding of Signals, in "Pattern-Directed Inference Systems", D. Waterman and F. Hayes-Roth (eds), Academic Press, New York, pp. 483-501, 1978
- O'Gorman, F., Clowes, M.B., Finding Picture Edges Through Collinearity of Feature Points, Proc. 3rd IJCAI, pp. 543-555, Stanford, CA, August 1973
- Ohlander, R., Analysis of Natural Scenes, Ph.D. Thesis, Carnegie-Mellon Univ., Pittsburgh, PA, April 1975
- Pentland, A., The Visual Inference of Shape: Computation from Local Features, Ph.D. Thesis, MIT, June 1982
- Perkins, W.A., Multilevel Vision Recognition System, Proc 3rd IJCPR, pp. 739-744, San Diego, CA, November 1976.

- Post, E., Formal Reductions of the General Combinatorial Problem, Am J. of Math, Vol 65, pp 197-268, 1943
- Pratt, W.K., "Digital Image Processing" New York, Wiley, pp 493-501, 1978
- Prewitt, J. M. S., Object Enhancement and Extraction, in "Picture Processing and Psychopictorics", B S Lipkin and A. Rosenfeld Eds, Academic Press, New York, pp. 75-149, 1970
- Ramer, E. U., The Transformation of Photographic Images into Stroke Arrays, IEEE Trans Circuits and Systems, Vol CAS-22, No 4, pp 363-374, April 1975
- Riseman, E. M., Hanson, A. R., Design of a Semantically Directed Vision Processor, COINS TR-73C-1, Computer and Information Sciences, Univ of Massachusetts, January 1974
- Riseman, E. M., Arbib, M. A., Survey Computational Techniques in the Visual Segmentation of Static Scenes, Computer Graphics and Image Processing, Vol 6, pp 221-276, 1977
- Roberts, L. G., Machine Perception of Three-Dimensional Solids, Technical Report 315, MIT Lincoln Laboratories, Lexington, MA, May 1963, Also in "Optical and Electro-Optical Information Processing", J. T. Tippett et al (eds), MIT Press, Cambridge, MA, pp 159-197, 1965
- Robinson, G. S., Edge detection by Compass Gradient Masks, Computer Graphics and Image Processing, Vol 6, No. 5, pp. 492-501, October 1977.
- Rosenfeld, A., Picture Processing by Computers, Academic Press, New York, 1969
- Rosenfeld, A., Thurston, M., Edge and Curve Detection for Visual Scene Analysis, IEEE Trans on Computers, Vol. C-20, No 5, pp 562-569, May 1971.
- Rosenfeld, A., Thurston, M., Lee, Y. H., Edge and Curve Detection: Further Experiments, IEEE Trans on Computers, Vol. C-21, pp. 677-714, 1972
- Rosenfeld, A., Hummel, R. A., Zucker, S. W., Scene Labelling by Relaxation Operations, IEEE Trans Systems, Man, and Cybernetics, SMC-6, p. 420-433, 1976
- Rosenfeld, A., Picture Processing: 1976, Computer Graphics and Image Processing, Vol. 6, No. 2, June 1977, pp. 157-183.

- Rosenfeld, A., Picture Processing 1977, Computer Graphics and Image Processing, Vol 7, No 2, June 1978, pp. 211-242
- Rosenfeld, A., Picture Processing 1978, Computer Graphics and Image Processing, Vol 9, No 4, April 1979, pp. 354-393
- Rosenfeld, A., Picture Processing 1979, Computer Graphics and Image Processing, Vol. 13, No 1, May 1980, pp 46-79
- Rosenfeld, A., Picture Processing 1981, Computer Graphics and Image Processing, Vol 19, No 1, May 1982, pp 35-75
- Rychener, M D., Knowledge-Based Expert Systems A Brief Bibliography, Technical Report, CMU-CS-81-127, Dept of Computer Science, Carnegie-Mellon Univ, June 1981
- Shaheen, S I., Image Segmentation and Interpretation Using a Knowledge Database, Ph D Thesis, Dept of Electrical Eng, McGill University, Montreal, Quebec, 1979
- Shirai, Y., Analyzing Intensity Arrays Using Knowledge About Scenes, in "The Psychology of Computer Vision", P Winston (ed), McGraw Hill, New York, 1975a
- Shirai, Y., Edge Finding, Segmentation of Edges and Recognition of Complex Objects, Proc. 4th IJCAI, pp 674-681, Tbilisi, Georgia, USSR, September 1975b
- Shortliffe, E., "Computer-Based Medical Consultations MYCIN, New York, Elsevier, 1976
- Sloan, K.R. Jr., World Model Driven Recognition of Natural Scenes, Ph.D. Thesis, University of Pennsylvania, The Moore School of Electrical Engineering, Philadelphia, PA, June 1977
- Tanimoto, S., Pavlidis, T., A Hierarchical Data Structure for Picture Processing, Computer Graphics and Image Processing 4, pp. 104-199, 1975.
- Tanimoto, S., An Iconic/Symbolic Data Structuring Scheme, in "Artificial Intelligence and Pattern Recognition", Academic Press, NY, 1977.
- Tenenbaum, J.M., On Locating Objects by their Distinguishing Features in Multi-Sensory Images, SRI Technical Note 84, AI Center, Stanford Research Institute, Stanford, CA, 1973.

- Tenenbaum, J.M., Barrow, H.G., Experiments in Interpretation-Guided Segmentation, Artificial Intelligence, Vol. 8, No 3, pp 241-274, 1977.
- Tsotsos, J.K., Knowledge of the Visual Process: Content, Form, and Use, Proc 6th IJCP, pp. 654-669, Munich, Germany, Oct 19-22, 1982.
- Tsuji, S., Tomita, F., A Structural Analyzer for a Class of Textures, Computer Graphics and Image Processing, Vol 2, pp. 216-231, 1973.
- Uhr, L., Layered Recognition Cone Networks that Process, Classify and Describe, IEEE Trans Computers, C-21, pp 758-768, 1972
- Waltz, D., Understanding Line Drawings of Scenes with Shadows, in "The Psychology of Computer Vision", P. Winston (ed), McGraw Hill, New York, 1975.
- Waterman, D.A., Adaptive Production Systems, Complex Information Processing Working Paper No 285, Dept. of Psychology, Carnegie-Mellon University, December 1974.
- Wertheimer, M., Laws of Organization in Perceptual Forms, in [Ellis, 1938]
- Wilson, H.R., Bergen, J.R., A Four Mechanism Model for Spatial Vision, Vision Research 19, pp. 19-32, 1979
- Winograd, T., Frame Representations and the Procedural/Declarative Controversy, in "Representation and Understanding", Borrow and Collins (eds.), Academic Press, 1975.
- Winston, P.H., Learning Structural Descriptions from Examples, Ph D. Thesis, MIT, 1970.
- Woodham, R.J., A Cooperative Algorithm for Determining Surface Orientation from a Single View, Proc. 5th IJCAI, pp 635-641, Boston, MA, 1977.
- Yager, R.R., Multiple Objective Decision-Making Using Fuzzy Sets, Int. J. of Man-Machine Studies, Vol. 9, pp. 375-382, 1977.
- Yakimovsky, Y., Feldman, J., A Semantics-Based Decision Theory Region Analyzer, Proc. 3rd IJCAI, pp. 580-588, Stanford, CA, 1973.
- Yasnoff, W.A., Mui, J.K., Bacus, J.W., Error Measures for Scene Segmentation, Pattern Recognition, Vol. 9, pp. 217-231, 1977.

- Youssef, Y.M., Quantification and Characterization of the Motion and Shape of a Moving Cell, Ph.D. Thesis, Dept. of Electrical Engineering, McGill University, Montreal, 1982.
- Zadeh, L.A., Fuzzy Sets, Information and Control, Vol. 8, pp 338-353, 1965.
- Zadeh, L.A., A Fuzzy Set Theoretic Interpretation of Linguistic Hedges, Memo-335, Electronic Research Lab., University of California, Berkley, 1972.
- Zadeh, L.A., Outline of a New Approach to the Analysis of Complex Systems and Decision Processes, IEEE Trans. Systems, Man, and Cybernetics, Vol. SMC-3, No. 1, pp. 28-44, January 1973.
- Zadeh, L.A., A Fuzzy-Algorithmic Approach to the Definition of Complex or Imprecise Concepts, Int. J. of Man-Machine Studies, Vol 8, pp 247-291, 1976.
- Zucker, S.W., Rosenfeld, A., Davis, L.S., General Purpose Models Expectations About the Unexpected, Proc. 4th IJCAI, Tbilisi, Georgia, USSR, pp. 716-721, September 1975.
- Zucker, S.W., Region Growing: Childhood and Adolescence, Computer Graphics and Image Processing 5, pp. 382-399, 1976.
- Zucker, S.W., Relaxation Labelling and the Reduction of Local Ambiguities, in "Pattern Recognition and Artificial Intelligence", C.H. Chen (ed.), Academic Press, New York, 1977.
- Zucker, S.W., Hummel, R.A., Rosenfeld, A., An Application of Relaxation Labelling to Line and Curve Enhancement, IEEE Trans. Computers, C-26, pp. 394-403 and 922-929, 1977.
- Zucker, S.W., Production Systems with Feedback, in "Pattern Directed Inference Systems", D. Waterman and F. Hayes-Roth (eds.), pp. 539-555, Academic Press, New York, 1978.
- Zucker, S.W., Computer Vision and Human Perception: An Essay on the Discovery of Constraints, Proc. 7th IJCAI, Vancouver, B.C., August 1981.

APPENDIX A

THE KNOWLEDGE RULES

A.1. AREA ANALYSIS RULES

Rule # 401 IF 1) HIGH_REG_SIZE
 2) LOW_REG_VARIANCE_1
 3) LOW_REG_VARIANCE_2
 4) LOW_REG_VARIANCE_3

 THEN 1) CREATE_SMOOTH_AREA

Rule # 402 IF 1) AREA_IS_SMOOTH
 2) NOT_VLOW_ADJACENCY
 3) NOT_LOW_REG_SIZE
 4) LOW_REG_VARIANCE_1
 5) LOW_REG_VARIANCE_2
 6) LOW_REG_VARIANCE_3

 THEN 1) ADD_TO_SMOOTH_AREA

Rule # 403 IF 1) AREA_IS_SMOOTH
 2) NOT_LOW_AREA_SIZE

 THEN 1) SAVE_SMOOTH_AREA

Rule # 501 IF 1) LOW_REG_SIZE

 THEN 1) CREATE_TEXTURE_AREA

Rule # 502 IF 1) AREA_IS_TEXTURE
 2) NOT_VLOW_ADJACENCY
 3) LOW_REG_SIZE

 THEN 1) ADD_TO_TEXTURE_AREA

Rule # 503 IF 1) AREA_IS_TEXTURE
 2) NOT_LOW_AREA_SIZE
 3) NOT_LOW_NUM_REGS_IN_AREA
 4) NOT_LOW_AREA_VARIANCE_1
 5) NOT_LOW_AREA_VARIANCE_2
 6) NOT_LOW_AREA_VARIANCE_3

 THEN 1) SAVE_TEXTURE_AREA

Rule # 601 IF 1) LINE_IS_LOOP
 2) NOT_LOW_LINE_LENGTH

 THEN 1) SAVE_BOUNDED_AREA

Rule # 602 IF 1) HIGH_LINE_LENGTH
 2) LOW_START_END_DISTANCE

 THEN 1) SAVE_BOUNDED_AREA

Rule # 603 IF 1) MEDIUM_LINE_LENGTH
 2) VLOW_START_END_DISTANCE

 THEN 1) SAVE_BOUNDED_AREA

Rule # 701 IF 1) AREA_IS_SMOOTH
 2) NOT_LOW_AREA_VARIANCE_1
 3) NOT_LOW_AREA_VARIANCE_2
 4) NOT_LOW_AREA_VARIANCE_3

 THEN 1) RELABEL_AREA_TEXTURE

Rule # 702 IF 1) AREA_IS_TEXTURE
 2) LOW_AREA_VARIANCE_1
 3) LOW_AREA_VARIANCE_2
 4) LOW_AREA_VARIANCE_3

 THEN 1) RELABEL_AREA_SMOOTH

Rule # 703 IF 1) AREA_IS_SMOOTH
 2) VLOW_NUM_REGS_IN_AREA
 3) LINES_ARE_ABSENT
 4) HIGH_REG_UNIFORMITY_1
 5) HIGH_REG_UNIFORMITY_2
 6) HIGH_REG_UNIFORMITY_3

 THEN 1) FREEZE_AREA

Rule # 704 IF 1) AREA_IS_TEXTURE
 2) HIGH_REG_CONTRAST_1
 3) HIGH_REG_CONTRAST_2
 4) HIGH_REG_CONTRAST_3
 5) HIGH_LINE_CONNECTIVITY

 THEN 1) FREEZE_AREA

Rule # 705 IF 1) AREA_IS_BOUNDED
 2) LINE_IS_CLOCKWISE
 3) ONE_REGION_RIGHT
 4) HIGH_REG_UNIFORMITY_1
 5) HIGH_REG_UNIFORMITY_2
 6) HIGH_REG_UNIFORMITY_3

 THEN 1) FREEZE_AREA

Rule # 706 IF 1) AREA_IS_BOUNDED
 2) LINE_IS_NOT_CLOCKWISE
 3) ONE_REGION_LEFT
 4) HIGH_REG_UNIFORMITY_1
 5) HIGH_REG_UNIFORMITY_2
 6) HIGH_REG_UNIFORMITY_3

 THEN 1) FREEZE_AREA

Rule # 707 IF 1) AREA_IS_SMOOTH
 2) HIGH_NUM_REGS_IN_AREA
 3) NOT_HIGH_AREA_SIZE
 4) HIGH_REG_CONTRAST_1
 5) HIGH_REG_CONTRAST_2
 6) HIGH_REG_CONTRAST_3

 THEN 1) DELETE_AREA

Rule # 708 IF 1) AREA_IS_TEXTURE
 2) LOW_NUM_REGS_IN_AREA
 3) NOT_HIGH_AREA_SIZE
 4) HIGH_REG_UNIFORMITY_1
 5) HIGH_REG_UNIFORMITY_2
 6) HIGH_REG_UNIFORMITY_3

 THEN 1) DELETE_AREA

Rule # 709 IF 1) AREA_IS_BOUNDED
 2) HIGH_NUM_LINES_IN_AREA
 3) HIGH_LINE_CONTRAST_1
 4) HIGH_LINE_CONTRAST_2
 5) HIGH_LINE_CONTRAST_3

 THEN 1) DELETE_AREA

A.2. REGION ANALYSIS RULES

Rule # 801 IF 1) LOW_DIFF_REG_FEATURE_1
 2) LOW_DIFF_REG_FEATURE_2
 3) LOW_DIFF_REG_FEATURE_3

 THEN 1) MERGE_REGIONS

Rule # 802 IF 1) VLOW_REG_SIZE
 2) HIGH_ADJACENCY
 3) NOT_HIGH_DIFF_REG_FEATURE_1
 4) NOT_HIGH_DIFF_REG_FEATURE_2
 5) NOT_HIGH_DIFF_REG_FEATURE_3

 THEN 1) MERGE_REGIONS

Rule # 803 IF 1) LOW_REG_SIZE
 2) HIGH_REG_GRADIENT
 3) NOT_HIGH_DIFF_REG_FEATURE_1
 4) NOT_HIGH_DIFF_REG_FEATURE_2
 5) NOT_HIGH_DIFF_REG_FEATURE_3

 THEN 1) MERGE_REGIONS

Rule # 804 IF 1) LOW_REG_SIZE
 2) VHIGH_ASPECT_RATIO
 3) NOT_HIGH_DIFF_REG_FEATURE_1
 4) NOT_HIGH_DIFF_REG_FEATURE_2
 5) NOT_HIGH_DIFF_REG_FEATURE_3

 THEN 1) MERGE_REGIONS

Rule # 805 IF 1) LOW_REG_SIZE
 2) VLOW_ASPECT_RATIO
 3) NOT_HIGH_DIFF_REG_FEATURE_1
 4) NOT_HIGH_DIFF_REG_FEATURE_2
 5) NOT_HIGH_DIFF_REG_FEATURE_3

 THEN 1) MERGE_REGIONS

Rule # 806 IF 1) LOW_REG_SIZE
 2) HIGH_NUM_ADJACENT
 3) NOT_HIGH_DIFF_REG_FEATURE_1
 4) NOT_HIGH_DIFF_REG_FEATURE_2
 5) NOT_HIGH_DIFF_REG_FEATURE_3
 6) NOT_LOW_REG_SIZE

 THEN 1) MERGE_REGIONS

Rule # 807 IF 1) VLOW_REG_SIZE
 2) VLOW_REG_SIZE
 3) NOT_HIGH_DIFF_REG_FEATURE_1
 4) NOT_HIGH_DIFF_REG_FEATURE_2
 5) NOT_HIGH_DIFF_REG_FEATURE_3

 THEN 1) MERGE_REGIONS

Rule # 808 IF 1) LOW_DIFF_REG_FEATURE_1
 2) LOW_DIFF_REG_FEATURE_2
 3) LOW_DIFF_REG_FEATURE_3
 4) LOW_DIFF_REG_VARIANCE_1
 5) LOW_DIFF_REG_VARIANCE_2
 6) LOW_DIFF_REG_VARIANCE_3

 THEN 1) MERGE_REGIONS

Rule # 809 IF 1) LOW_DIFF_REG_FEATURE_1
 2) LOW_DIFF_REG_FEATURE_2
 3) LOW_DIFF_REG_FEATURE_3
 4) LOW_VALUE_BETWEEN

 THEN 1) MERGE_REGIONS

Rule # 901 IF 1) HISTOGRAM_IS_BIMODAL

 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 902 IF 1) NOT_LOW_HIST_BIMODALITY

 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 903 IF 1) HIGH_REG_SIZE
 2) HIGH_HIST_BIMODALITY

 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 904 IF 1) MEDIUM_REG_SIZE
 2) NOT_LOW_HIST_BIMODALITY
 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 905 IF 1) HIGH_REG_VARIANCE_1
 2) HIGH_REG_VARIANCE_2
 3) HIGH_REG_VARIANCE_3
 4) HISTOGRAM_IS_BIMODAL
 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 906 IF 1) HIGH_REG_SIZE
 2) NOT_LOW_REG_GRADIENT
 3) HISTOGRAM_IS_BIMODAL
 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 907 IF 1) NOT_LOW_REG_SIZE
 2) HIGH_REG_GRADIENT
 3) HISTOGRAM_IS_BIMODAL
 THEN 1) SPLIT_REGION_HISTOGRAM

Rule # 908 IF 1) NOT_LOW_REG_SIZE
 2) REGION_IS_BISECTED
 3) NOT_LOW_LINE_LENGTH
 4) HIGH_LINE_GRADIENT
 THEN 1) SPLIT_REGION_AT_LINES

A.3. LINE ANALYSIS RULES

Rule # 1401 IF 1) VLOW_LINE_LENGTH
 2) LOW_LINE_GRADIENT
 THEN 1) DELETE_LINE

Rule # 1402 IF 1) LOW_LINE_LENGTH
 2) VLOW_LINE_GRADIENT
 THEN 1) DELETE_LINE

Rule # 1403 IF 1) LINE_IS_OPEN
 2) LOW_LINE_LENGTH
 3) LOW_LINE_GRADIENT

 THEN 1) DELETE_LINE

Rule # 1404 IF 1) LOWER_LINE_LENGTH
 2) LOW_DIFF_LINE_GRADIENT
 3) VLOW_PARALLEL_DISTANCE

 THEN 1) DELETE_LINE

Rule # 1405 IF 1) LINE_IS_OPEN
 2) LOW_LINE_GRADIENT
 3) SAME_REGL1_REGR1

 THEN 1) DELETE_LINE

Rule # 1406 IF 1) LOW_LINE_GRADIENT
 2) NOT_SAME_REGL1_REGR1
 3) LOW_DIFF_REG_FEATURE_1
 4) LOW_DIFF_REG_FEATURE_2
 5) LOW_DIFF_REG_FEATURE_3

 THEN 1) DELETE_LINE

Rule # 1501 IF 1) LINE_END_IS_OPEN
 2) LOW_FRONT_DISTANCE

 THEN 1) JOIN_LINES_FORWARD

0

Rule # 1502 IF 1) LINE_END_IS_OPEN
 2) NOT_HIGH_FRONT_DISTANCE
 3) NOT_LOW_LINE_GRADIENT

 THEN 1) JOIN_LINES_FORWARD

Rule # 1503 IF 1) LINE_END_IS_OPEN
 2) NOT_HIGH_FRONT_DISTANCE
 3) EQUAL_LINE_DIRECTION

 THEN 1) JOIN_LINES_FORWARD

Rule # 1504. IF 1) LINE_END_IS_OPEN
 2) NOT_VLOW_LINE_GRADIENT
 3) NOT_VHIGH_FRONT_DISTANCE
 4) SAME_REGL1_P EGL2
 5) SAME_REGR1_REGR2

 THEN 1) JOIN_LINES_FORWARD

Rule # 1505 IF 1) LINE_END_IS_OPEN
 2) NOT_VLOW_LINE_GRADIENT
 3) NOT_VHIGH_FRONT_DISTANCE
 4) SAME_REGL1_REGR2
 5) SAME_REGR1_REGR2

 THEN 1) JOIN_LINES_FORWARD

Rule # 1506 IF 1) LINE_END_IS_OPEN
 2) NOT_VLOW_LINE_GRADIENT
 3) NOT_VHIGH_FRONT_DISTANCE
 4) SAME_REGL1_REGL2
 5) SAME_REGR1_REGL2

 THEN 1) JOIN_LINES_FORWARD

Rule # 1507 IF 1) LINE_END_IS_OPEN
 2) HIGH_LINE_GRADIENT

 THEN 1) EXTEND_LINE_FORWARD

Rule # 1508 IF 1) LINE_END_IS_OPEN
 2) NOT_LOW_LINE_GRADIENT
 3) NOT_LOW_LINE_LENGTH

 THEN 1) EXTEND_LINE_FORWARD

Rule # 1509 IF 1) LINE_END_IS_OPEN
 2) NOT_LOW_LINE_GRADIENT
 3) NOT_LOW_NUM_REGS_LEFT
 4) NOT_LOW_NUM_REGS_RIGHT

 THEN 1) EXTEND_LINE_FORWARD

Rule # 1510 IF 1) LINE_END_IS_OPEN
 2) NOT_LOW_LINE_LENGTH
 3) ONE_REGION_LEFT
 4) ONE_REGION_RIGHT
 5) NOT_SAME_REGL1_REGR1

 THEN 1) INSERT_LINE_FORWARD

Rule # 1511

IF 1) LINE_END_IS_OPEN
2) NOT_LOW_LINE_GRADIENT
3) LOW_NUM_REGS_LEFT
4) LOW_NUM_REGS_RIGHT
5) NOT_SAME_REGL1_REGR1

THEN 1) INSERT_LINE_FORWARD

Rule # 1601

IF 1) LINE_START_IS_OPEN
2) LOW_HIND_DISTANCE

THEN 1) JOIN_LINES_BACKWARD

Rule # 1602

IF 1) LINE_START_IS_OPEN
2) NOT_HIGH_HIND_DISTANCE
3) NOT_LOW_LINE_GRADIENT

THEN 1) JOIN_LINES_BACKWARD

Rule # 1603

IF 1) LINE_START_IS_OPEN
2) NOT_HIGH_HIND_DISTANCE
3) EQUAL_LINE_DIRECTION

THEN 1) JOIN_LINES_BACKWARD

Rule # 1604

IF 1) LINE_START_IS_OPEN
2) NOT_VLOW_LINE_GRADIENT
3) NOT_VHIGH_HIND_DISTANCE
4) SAME_REGL1_REGL2
5) SAME_REGR1_REGR2

THEN 1) JOIN_LINES_BACKWARD

Rule # 1605

IF 1) LINE_START_IS_OPEN
2) NOT_VLOW_LINE_GRADIENT
3) NOT_VHIGH_HIND_DISTANCE
4) SAME_REGL1_REGR2
5) SAME_REGR1_REGR2

THEN 1) JOIN_LINES_BACKWARD

Rule # 1606

IF 1) LINE_START_IS_OPEN
2) NOT_VLOW_LINE_GRADIENT
3) NOT_VHIGH_HIND_DISTANCE
4) SAME_REGL1_REGL2
5) SAME_REGR1_REGL2

THEN 1) JOIN_LINES_BACKWARD

```

Rule # 1607          IF    1) LINE_START_IS_OPEN
                       2) HIGH_LINE_GRADIENT

                       THEN 1) EXTEND_LINE_BACKWARD
                           /

Rule # 1608          IF    1) LINE_START_IS_OPEN
                       2) NOT_LOW_LINE_GRADIENT
                       3) NOT_LOW_LINE_LENGTH

                       THEN 1) EXTEND_LINE_BACKWARD

Rule # 1609          IF    1) LINE_START_IS_OPEN
                       2) NOT_LOW_LINE_GRADIENT
                       3) NOT_LOW_NUM_REGS_LEFT
                       4) NOT_LOW_NUM_REGS_RIGHT

                       THEN 1) EXTEND_LINE_BACKWARD

Rule # 1610          IF    1) LINE_START_IS_OPEN
                       2) NOT_LOW_LINE_LENGTH
                       3) ONE_REGION_LEFT
                       4) ONE_REGION_RIGHT
                       5) NOT_SAME_REGL1_REGR1

                       THEN 1) INSERT_LINE_BACKWARD

Rule # 1611          IF    1) LINE_START_IS_OPEN
                       2) NOT_LOW_LINE_GRADIENT
                       3) LOW_NUM_REGS_LEFT
                       4) LOW_NUM_REGS_RIGHT
                       5) NOT_SAME_REGL1_REGR1

                       THEN 1) INSERT_LINE_BACKWARD

Rule # 1701          IF    1) NOT_LOW_LINE_LENGTH
                       2) LOW_LINE_LENGTH
                       3) LINES_ARE_TOUCHING
                       4) LOW_FRONT_POINT

                       THEN 1) CONNECT_LINES_FORWARD

Rule # 1702          IF    1) NOT_LOW_LINE_LENGTH
                       2) NOT_LOW_LINE_LENGTH
                       3) LINES_ARE_TOUCHING
                       4) VLOW_FRONT_POINT

                       THEN 1) CONNECT_LINES_FORWARD

```

Rule # 1703

IF 1) LINES_ARE_TOUCHING
2) NOT_HIGH_FRONT_POINT
3) HIGH_LINEF_GRAD_VAR
4) LOW_DIFF_LINE_FEATURE_1
5) LOW_DIFF_LINE_FEATURE_2
6) LOW_DIFF_LINE_FEATURE_3

THEN 1) CONNECT_LINES_FORWARD

Rule # 1704

IF 1) LINES_ARE_TOUCHING
2) MEDIUM_FRONT_POINT
3) EQUAL_LINE_END_DIRECTION
4) NOT_EQUAL_LINEF_STARTEND_DIR

THEN 1) CONNECT_LINES_FORWARD

Rule # 1705

IF 1) LINES_ARE_TOUCHING
2) NOT_LOW_LINEF_LENGTH
3) EQUAL_LINE_STARTEND_DIR
4) EQUAL_LINEF_STARTEND_DIR
5) LOW_FRONT_POINT
6) VLOW_DIFF_LINE_DIRECTION

THEN 1) CONNECT_LINES_FORWARD

Rule # 1801

IF 1) NOT_LOW_LINE_LENGTH
2) LOW_LINEB_LENGTH
3) LINES_ARE_TOUCHING
4) LOW_HIND_POINT

THEN 1) CONNECT_LINES_FORWARD

Rule # 1802

IF 1) NOT_LOW_LINE_LENGTH
2) NOT_LOW_LINEB_LENGTH
3) LINES_ARE_TOUCHING
4) VHIGH_HIND_POINT

THEN 1) CONNECT_LINES_BACKWARD

Rule # 1803

IF 1) LINES_ARE_TOUCHING
2) NOT_LOW_HIND_POINT
3) HIGH_LINEB_GRAD_VAR
4) LOW_DIFF_LINE_FEATURE_1
5) LOW_DIFF_LINE_FEATURE_2
6) LOW_DIFF_LINE_FEATURE_3

THEN 1) CONNECT_LINES_BACKWARD

Rule # 1804

IF 1) LINES_ARE_TOUCHING
2) MEDIUM_HIND_POINT
3) EQUAL_LINE_START_DIRECTION
4) NOT_EQUAL_LINEB_STARTEND_DIR

THEN 1) CONNECT_LINES_FORWARD

Rule # 1805

IF 1) LINES_ARE_TOUCHING
2) NOT_LOW_LINEB_LENGTH
3) EQUAL_LINE_STARTEND_DIR
4) EQUAL_LINEB_STARTEND_DIR
5) HIGH_HIND_POINT
6) VLOW_DIFF_LINE_DIRECTION

THEN 1) CONNECT_LINES_BACKWARD

APPENDIX B

THE CONTROL RULES

B.1. FOCUS OF ATTENTION RULES

Rule #101 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE
 (3) NEXT_PREV_PROCESS_WAS_NOT_FOCUS

 THEN (1) GET_NEXT_AREA

Rule #102 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE
 (3) NEXT_PREV_PROCESS_WAS_FOCUS
 (4) NEXT_PREV_PROCESS_WAS_ACTIVE

 THEN (1) GET_NEXT_AREA

Rule #103 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE
 (3) NEXT_PREV_PROCESS_WAS_FOCUS
 (4) NEXT_PREV_PROCESS_WAS_NOT_ACTIVE

 THEN (1) CLEAR_REGION_LIST
 (2) CLEAR_LINE_LIST

Rule #104 IF (1) FOCUS_IS_ON_AREAS
 (2) AREAS_ARE_ABSENT

 THEN (1) DEFOCUS

Rule #105 : IF (1) PREV_PROCESS_WAS_NOT_FOCUS

 THEN (1) GET_NEXT_REGION_LABEL

✓ Rule #106 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_HIGHEST_ADJACENT_REGION

Rule #107 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_LARGEST_REGION

Rule #108 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_SMALLEST_REGION

Rule #109 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_LOWEST_ADJACENT_REGION

Rule #110 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_LINE_LABEL

Rule #111 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_NEAR_LINE

Rule #112 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_STRONGEST_LINE

Rule #113 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_WEAKEST_LINE

Rule #114 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_LONGEST_LINE

Rule #115 IF (1) PREV_PROCESS_WAS_NOT_FOCUS
 THEN (1) GET_NEXT_SHORTEST_LINE

Rule #201 IF (1) HIGH_LINE_GRADIENT
 (2) HIGH_LINE_LENGTH
 (3) SAME_REGL1_REGR1
 THEN (1) GET_NEXT_REGION_LEFT

Rule #202 IF (1) LINE_IS_LOOP
 (2) LINE_IS_CLOCKWISE
 (3) HIGH_PEGR_ADJACENCY

 THEN (1) GET_NEXT_REGION_RIGHT

Rule #203 IF (1) LINE_IS_LOOP
 (2) LINE_IS_NOT_CLOCKWISE
 (3) HIGH_REGL_ADJACENCY

 THEN (1) GET_NEXT_REGION_LEFT

Rule #204 IF (1) HIGH_REG_SIZE
 (2) HIGH_REG_GRADIENT

 THEN (1) GET_NEXT_TOUCHING_LINE

Rule #205 IF (1) NOT_VLOW_PARALLEL_DISTANCE

 THEN (1) GET_NEXT_PARALLEL_LINE

B. 2. THE META-RULES

Meta-Rule # 1 IF (1) REGIONS_ARE_ABSENT

 THEN (1) INITIALIZE_REGIONS

Meta-Rule # 2 IF (1) LINES_ARE_ABSENT

 THEN (1) INITIALIZE_LINES

Meta-Rule # 3 IF (1) AREAS_ARE_ABSENT

 THEN (1) GENERATE_AREAS

Meta-Rule # 4 IF (1) PREV_PROCESS_WAS_GENAREAS
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) GENERATE_AREAS

Meta-Rule # 5 IF (1) PREV_PROCESS_WAS_GENAREAS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule # 6 IF (1) PROCESS_WAS_FOCUS
 (2) PROCESS_WAS_NOT_ACTIVE
 (3) NEXT_PREV_PROCESS_WAS_FOCUS
 (4) NEXT_PREV_PROCESS_WAS_NOT_ACTIVE

 THEN (1) STOP

Meta-Rule # 7 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE
 (3) NEXT_PREV_PROCESS_WAS_NOT_FOCUS

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule # 8 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE
 (3) NEXT_PREV_PROCESS_WAS_FOCUS
 (4) NEXT_PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule # 9 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_REGION_RULES

Meta-Rule #10 IF (1) PREV_PROCESS_WAS_REGIONS
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_REGION_RULES

Meta-Rule #11 IF (1) PREV_PROCESS_WAS_REGIONS
 (2) PREV_PROCESS_WAS_NOT_ACTIVE

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule #12 IF (1) PREV_PROCESS_WAS_REGIONS
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule #13 IF (1) PREV_PROCESS_WAS_NOT_FOCUS

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule #14 IF (1) PREV_PROCESS_WAS_REGIONS

 THEN (1) MATCH_LINE_RULES

Meta-Rule #15 IF (1) PREV_PROCESS_WAS_LINES
 (2) PREV_PROCESS_WAS_NOT_ACTIVE

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule #16 IF (1) PREV_PROCESS_WAS_LINES
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_FOCUS_RULES

Meta-Rule #17 IF (1) PREV_PROCESS_WAS_LINES
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_LINE_RULES

Meta-Rule #18 IF (1) PREV_PROCESS_WAS_FOCUS
 (2) PREV_PROCESS_WAS_ACTIVE

 THEN (1) MATCH_LINE_RULES

Meta-Rule #19 IF (1) PREV_PROCESS_WAS_LINES

 THEN (1) MATCH_REGION_RULES

APPENDIX C

THE STRATEGY RULES

In chapter 6, the strategy selection process was introduced as a fuzzy decision-maker. This appendix describes how the constraints that control this process were implemented in the form of strategy rules. These are condition-action pairs similar to the knowledge and control rules listed in appendices A and B. Their conditions are matched on the performance measures, and their actions assign states to each of the strategy elements. The actions of the strategy rules are

Set REGION PATH STRATEGY to HIGHEST ADJACENT
Set REGION PATH STRATEGY to LOWEST ADJACENT
Set REGION PATH STRATEGY to LARGEST SIZE
Set REGION PATH STRATEGY to SMALLEST SIZE

Set LINE PATH STRATEGY to NEAREST LINE
Set LINE PATH STRATEGY to LONGEST LINE
Set LINE PATH STRATEGY to SHORTEST LINE
Set LINE PATH STRATEGY to STRONGEST LINE
Set LINE PATH STRATEGY to WEAKEST LINE

Set UPDATING STRATEGY to SUCCESSIVE
Set UPDATING STRATEGY to SINGLE

Set PROCESS PRIORITY to REGIONS
Set PROCESS PRIORITY to LINES

Set REGION RULE PRIORITY to MERGING
Set REGION RULE PRIORITY to SPLITTING

Set LINE RULE PRIORITY to ADDITION
Set LINE RULE PRIORITY to DELETION

The conditions of these rules are formulated according to the entries in tables (6 2) and (6 3). These describe the relation between each strategy element and all the performance measures. Following is a step-by-step description of how the rules are created.

(1) For each strategy element, a number of rules are formulated that is equal to the number of states that an element can assume. Therefore, a rule is defined by its action (as listed above) which sets a strategy element to a specific state.

(2) Each rule has a number of conditions equal to the number of performance measures that can affect the strategy element which is set by this rule. (These are the measures that have + or - in table (6 2) for that strategy element). Therefore, a condition in a rule corresponds to the effect of one performance measure on the strategy element.

(3) The feature in each condition is the performance measure whose entry in table (6 2) is being considered.

(4) The + and - entries in table (6 2) select one of the two basic operators HIGH and LOW, respectively.

(5) The entries in table (6 3) determine a "symbolic hedge" [Zadeh, 1972] to be attached to the operator in (3) as follows:

VL	- SLIGHTLY
L	- PARTIALLY
M	- MODERATELY
H	- SIGNIFICANTLY
VH	- EXTREMELY

(6) Finally, the set of strategy rules is repeated for all types of areas. The symbolic hedges vary according to area type in table (6.3). Therefore, a different rule must be included for each of the three area types.

The procedure is best described by the following example. The rules that set the updating strategy for smooth areas are

```

IF  (1) The AREA is SMOOTH
     (2) The REGION UNIFORMITY is EXTREMELY HIGH
     (3) The REGION CONTRAST is SIGNIFICANTLY HIGH
     (4) The LINE CONTRAST is SIGNIFICANTLY HIGH
     (5) The LINE CONNECTIVITY is MODERATELY LOW
     (6) The NUMBER OF REGIONS is MODERATELY LOW
     (7) The NUMBER OF LINES is MODERATELY LOW

THEN (1) Set UPDATING STRATEGY to SUCCESSIVE

IF  (1) The AREA is SMOOTH
     (2) The REGION UNIFORMITY is EXTREMELY LOW
     (3) The REGION CONTRAST is SIGNIFICANTLY LOW
     (4) The LINE CONTRAST is SIGNIFICANTLY LOW
     (5) The LINE CONNECTIVITY is MODERATELY HIGH
     (6) The NUMBER OF REGIONS is MODERATELY HIGH
     (7) The NUMBER OF LINES is MODERATELY HIGH

THEN (1) Set UPDATING STRATEGY to SINGLE

```

Note that two other sets of rules are generated by changing condition (1) above to account for the two other area types. The entire set of strategy rules can be generated by following the same procedure.

Finally, the reader is reminded that the evaluation of each condition results in a fuzzy variable. This corresponds to a level of satisfaction with that condition. The logical AND on all conditions corresponds to the selection of the

condition with the lowest satisfaction level. The rule with highest such value is selected to execute its action. This corresponds to the mini-max operation of the fuzzy decision-maker.

+ * +