# A COOPERATIVE SCHEME FOR IMAGE UNDERSTANDING USING MULTIPLE SOURCES OF INFORMATION

by

JAY GLICKSMAN

Technical Report TN 82-13
November 1982

Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1W5

This report was submitted as a thesis in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

# A COOPERATIVE SCHEME FOR IMAGE UNDERSTANDING USING MULTIPLE SOURCES OF INFORMATION

by

JAY GLICKSMAN

B.Sc. The University of Toronto, 1975
M.E. The University of Utah, 1977

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

DEPARTMENT OF COMPUTER SCIENCE

We accept this thesis as conforming

to the required standard

.....Alan Mackworth.....

.....[illegible].....

THE UNIVERSITY OF BRITISH COLUMBIA

November 1982

## Abstract

One method of resolving the ambiguity inherent in interpreting images is to add different sources of information. The multiple information source paradigm emphasizes the ability to utilize knowledge gained from one source that may not be present in another. However, utilizing disparate information may create situations in which data from different sources are inconsistent.

A schemata-based system has been developed that can take advantage of multiple sources of information. Schemata are combined into a semantic network via the relations decomposition, specialization, instance of, and neighbour. Control depends on the structure of the evolving network and a cycle of perception. Schemata cooperate by message passing so that attention can be directed where it will be most advantageous.

This system has been implemented to interpret aerial photographs of small urban scenes. Geographic features are identified using up to three information sources: the intensity image, a sketch map, and information provided by the user. The product is a robust system where the accuracy of the results reflects the quality and amount of data provided. Images of several geographic locales are analyzed, and positive results are reported.

Table of Contents

List of Tables

## List of Figures

## Acknowledgments

First and foremost, I wish to thank my supervisor, Alan Mackworth, for his support (both intellectual and financial), guidance, and advice. I am also indebted to him for writing Mapsee and exposing me to the potential for exploiting the constraints inherent in models of geographic objects. I would also like to acknowledge his permission to use the diagrams that are found in Figures 3.3 and 3.4.

Thanks are also extended to the rest of my committee, Bob Woodham, Richard Rosenberg, David Kirkpatrick, and Anne Treisman, who made helpful comments along the way to this final document.

I am grateful to Bill Havens, Jan Mulder, and Alan for explanations of the inner workings of Maya and Mapsee2, and for fixing the bugs that I found. Jim Little gave me a great deal of assistance with algorithms for line generalization, finding channels and passes, and edge detection using a Sobel edge detector. The latter program was superceded by the Marr-Hildreth edge detector (so will not be described herein) but that was not Jim's fault.

I would like to thank my officemates for being interested in what I was doing and for not being visionaries which forced me to explain my work in English. I learned a lot about what I had done from those discussions with Randy Goebel and Bob Mercer.

Many members of the Laboratory for Computational Vision provided inspiration and feedback. Especially helpful were Jim Little, Jan Mulder, Roger Browse, Alan Mackworth, Bill Havens, Bob Woodham, and Marc Majka.

And in a class all by herself, Susan Suh. Thanks.

# CHAPTER 1

# Introduction

## 1.1. The Domain

Remote sensing is perception at a distance. Like computer vision, it is primarily concerned with the identification and representation of objects in the external world. Computer-based remote sensing and computer vision both have as their usual source of information a two-dimensional array of intensity values. Unlike the general vision problem, computer-based remote sensing cannot take advantage of many additional three-dimensional cues. Furthermore, whereas human vision (the simulation of which is often typified as the vision problem) is restricted to a narrow range of the electromagnetic spectrum, remote sensing often utilizes several different portions of the spectrum.

Figure 1.1 was taken from an airplane flying over Ashcroft, British Columbia. People have no problem discerning geographic entities such as the roads, rivers, buildings, or railway tracks. Aerial photo-interpretation is quite useful for such tasks as land management, cartography, monitoring pollution, erosion, and urban growth, estimating crop yield, and tracking the movement of ships, cars, and the like.

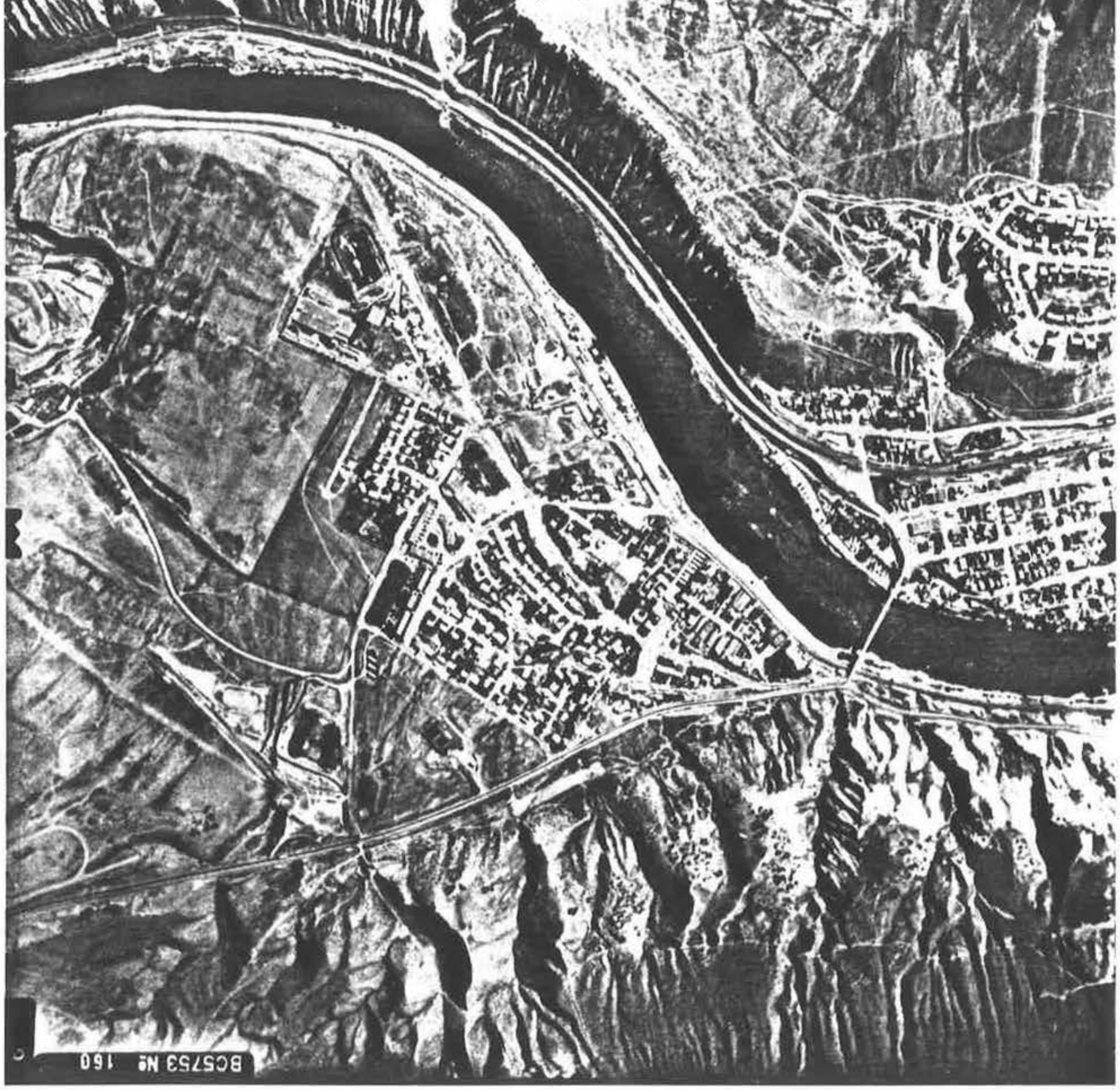


Figure 1.1 An Aerial Photograph
(scale reduced from original 23 cm. x 23 cm. photograph)

Computer-based aerial photo-interpretation is also a useful restriction of the machine vision problem because it is essentially two-dimensional and the number of objects that are considered is manageable. However, as has been found to be generally true in Artificial Intelligence, tasks that seem "easy" for people, like language and vision, are the most difficult to do by machine.

What salient features of an aerial photograph such as Figure 1.1 enable a person to identify its parts? Certainly the intensities of the various regions and where the intensities change are important. But that is rarely sufficient. Generally, we must use our knowledge of the objects themselves--their shapes, relative sizes, and the configurations of many objects taken together.

This dissertation discusses the use of knowledge of the objects to be modelled in geographic scenes. It aims to combine the information that can be extracted directly from images with data dependent on the domain of the task. Model knowledge can be used to organize the data and to control the process of interpretation.

## 1.2. Steps Toward a Solution

The information found in a single, monochromatic image such as Figure 1.1 is often very ambiguous. The transition from a bright to a dark region might represent the boundary between a bridge and water, a road and a building, the sunlit and shadowed parts of a ridge, or many other pairs of objects. While some aerial photographs can be readily interpreted by a novice, others are so ambiguous that even experts may have difficulty in discovering the correct interpretation.

One way to aid the interpretation of an image, for both people and a computer program, is to add more information. Maps (e.g. topographic and road maps) contain many stylized symbols which make it easier to understand them. A sketch map is a freehand drawing of the crucial features of an area. It encodes a great deal of information in a very sparse picture (see Figure 1.2). Easily drawn, it can be used to aid the interpretation of a corresponding image.

The work described in this document is primarily concerned with the usefulness of combining different types of information and the mechanisms to do so. A third source of information that can be utilized, besides the intensity image and a sketch map, is the user who can provide general information (e.g. "The nominal scale of the image is 1:10000") or very specific information (e.g. "There is a bridge in the centre of the image"). Also, two features that can be derived from an intensity image are edges and regions. They are generally extracted in different ways and can be used to emphasize different aspects of the image.

All these different sorts of data must be related to the models of the objects to be identified in the scene. However,

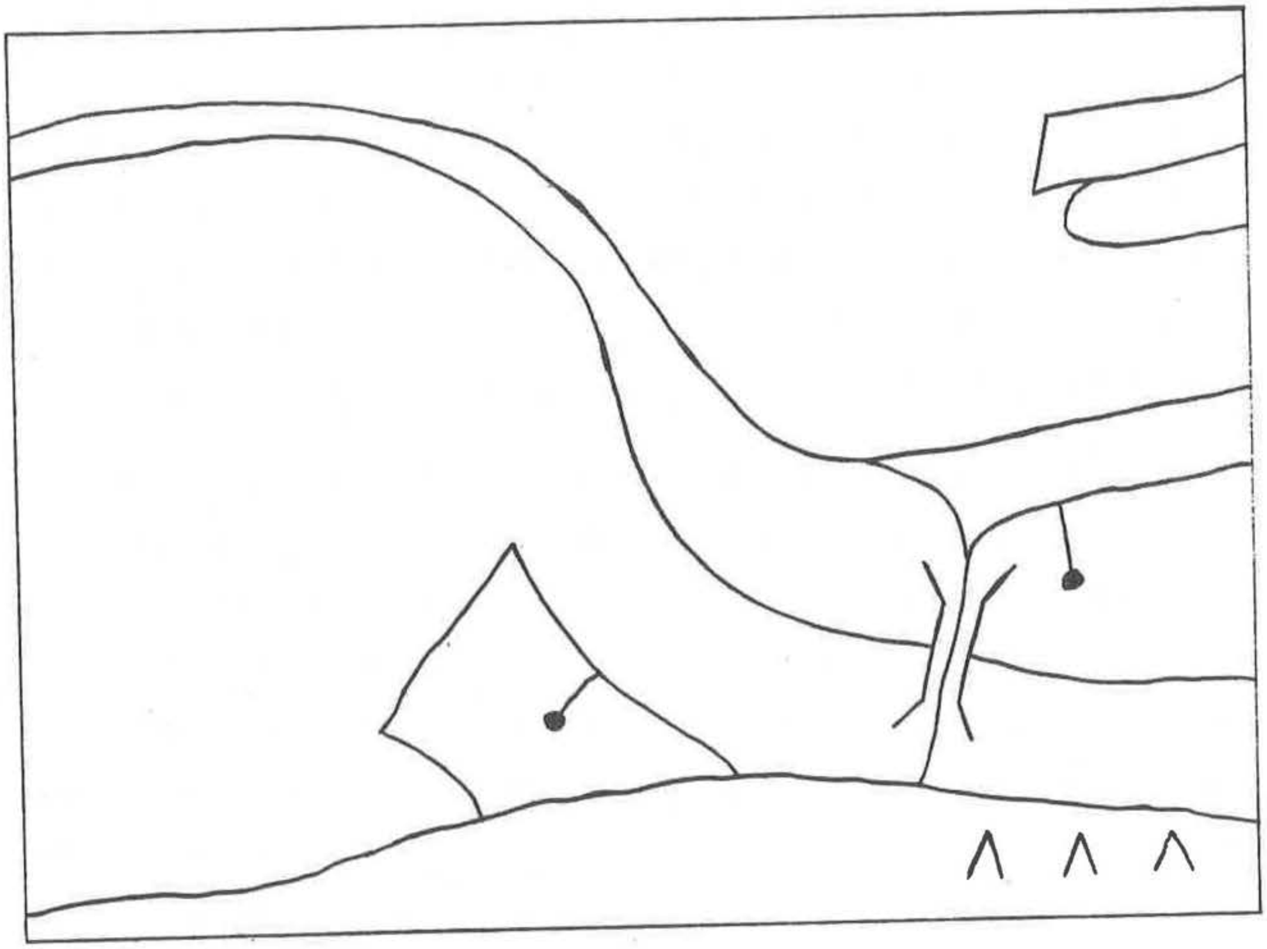


Figure 1.2 A Sketch Map

while it is hoped that all the evidence will support the same conclusion, mismatches may occur between the data and the models. These difficulties must be handled in a consistent and robust manner so that the accuracy of the results will reflect the quality of the available data.

One way of accomodating disparate types of information is to relate them all to the objects themselves. Schemata, or frames, are a data structure that can be used to store all the facts pertinent to a particular entity such as a road. Schemata contain a declarative part where information derived from an interpretation can be stored (e.g. "The edges that correspond to this road in the image are 11, 12, and 13") and a procedural part that contains methods for utilizing information sources to instantiate the existence of the object (such as using the sketch map with the image, using the image alone).

Schemata can be conjoined into a network that reflects their organization into cohesive units in the scene. For instance, a network of roads and bridges can be related as could all the entities that are part of a landmass delineated by a body of water. Agreement in the interpretation of neighbouring instances raises the confidence of the entire group. For example, roads are often found next to buildings, not ships. Furthermore, the existence of some instances raises the expectation that there will be other related instances nearby. Identification of a bridge points to the possibility of a river flowing under it. Using model knowledge in this way is an island-driven control strategy wherein the discovery of one object leads to others nearby. This group of objects expands in subsequent steps to encompass all of the relevant parts of the scene.

It is crucial that the system be procedurally adequate so that objects are not incorrectly instantiated. The procedures attached to schemata should not presuppose the existence of an object without finding the proper support in the data nor should they jump to conclusions. This is intimately related to the descriptive adequacy of the models--are definitions sufficient to confirm or deny the existence of the objects using the available sources of information? The descriptions must fit into a midrange, neither too limited nor too encompassing. In addition, the descriptions must be suitable for any possible applications that are of interest (e.g. "How long are the bridges in an image").

## 1.3. The Scope of the Thesis

Several disparate information sources are combined to interpret aerial photographs. Along with the digitized image, information from sketch maps and the user is made available. The interpretation results depend on the amount and quality of the information provided.

A new variety of schemata has been developed along with a system for manipulating them. The schemata provide a consistent, uniform repository for information coming from different information sources. Schemata contain attached procedures which control the interpretation process. As entities are discovered in the image they cause instances of schemata to be created and linked into a semantic network. Traversal of the growing network enables control to be either top-down or bottom-up, whichever is most appropriate.

The nodes of the semantic network communicate by passing messages. The messages are suggestions for future instantiations and are placed on a priority queue. Priorities depend upon the confidence values of the instances sending the messages and are determined during instantiation by how well the data fit the models. Processing proceeds in a cyclic manner following a cycle of perception. This cycle provides convenient points where interaction with the user can take place.

Several computational tools are developed to aid in the instantiation of geographic entities. Cluster analysis using Guassian-smoothed histograms is an inference mechanism for determining which data elements best fit model definitions. A method of relaxing thresholds is based on the used of model knowledge. In addition, many thresholds can be reduced to single values when model knowledge is combined with global schemata.

In summary, this dissertation describes a vision system that can accomodate information from several sources. It is object-oriented and based around schemata that form a semantic network of geographic objects and instances of those objects. It utilizes the current context of instantiated entities to control where future effort should be directed.

## 1.4. Reading Guide

The next three chapters correspond to the three parts of the title of the thesis in reverse order. Chapter 2 introduces the multiple information sources paradigm as the underlying hypothesis of this work. Chapter 3 is concerned with one aspect of image understanding: model-based visual perception. Previous relevant research in the field is outlined. In addition, three criteria for judging the success of a model-based vision system are described. Chapter 4 discusses the cooperative scheme, a schema-based system that is used to combine multiple information sources. Each section of this chapter describes a component of the system.

Chapter 5 discusses MISSEE, the implementation of the system described in Chapter 4. Results and evaluation of the system are the topic of Chapter 6. Finally, Chapter 7 summarizes the work and points to possible directions future research might proceed in.

Readers who are concerned with issues of knowledge representation and not vision may wish to read only those parts of the dissertation concerned with schemata. The relevant sections are 3.3, 3.4, 4.2, 4.5.3, 5.2, 5.4.2, 5.6, 6.6, 6.7, and Appendix A.

# CHAPTER 2

# Multiple Information Sources

## 2.1. A New Paradigm

### 2.1.1. Paradigms

In his influential book, "The Structure of Scientific Revolutions" (Kuhn, 1970), Thomas Kuhn presents a theory to account for the means by which science progresses in a particular field. "Normal science" generally revolves around the notion of a "paradigm". Paradigms "provide models from which spring particularly coherent traditions of scientific research" (p. 10). Where paradigms exist (e.g. Ptolemaic astronomy) advances in the field consist of sorting out the problems not solved by the work that defines the paradigm. It is important to note that the paradigm influences which problems are significant. Thus scientists not working within the paradigm are not considered to be working on meaningful problems by the adherents of the currently dominant paradigm.

Paradigms can be adopted in two ways. An existing paradigm can be supplanted by a new theory, the "so-called revolution", which redefines legitimate problems and research methods (e.g. the Copernican "revolution" which replaced Ptolemaic astronomy).

> [The new scientific] achievement [must be] sufficiently unprecedented to attract an enduring group of adherents away from competing modes of scientific activity. Simul-

> taneously, it [must be] sufficiently open-ended to leave all sorts of problems for the redefined group of practitioners to resolve. (p. 10)

Paradigms can also emerge in a young science where none are currently established. In pre-paradigmatic science, there are no generally accepted views which act as a focus for research. Rather, there are many small groups of practitioners with their own views and methodologies. In this state, science proceeds more slowly:

> . . . when the individual scientist can take a paradigm for granted, he need no longer, in his major works, attempt to build his field anew, starting from first principles and justifying the use of each concept introduced. That can be left to the writer of textbooks. (pp. 19-20)

Only when a paradigm has been established can progress become cumulative so that one can stand on the shoulders of giants (as Newton wrote) rather than on their toes.

## 2.1.2. The Current State of Computational Vision

Artificial Intelligence, in general, and Computational Vision, in particular, are very young sciences without a clearly recognized central paradigm. For exactly this reason, some people question whether they are even sciences at all. Compared to physics, the lack of unifying principles is noteworthy[1].

[1]Unfortunately there are many older disciplines that have the same difficulty. This leads some to question whether Kuhn's theory is applicable to everything that is labelled science. It leads others to question whether the label "science" is always appropriately applied.

While several textbooks on Artificial Intelligence have been written (Nilsson, 1971 and 1980; Winston, 1977), only recently has a textbook on computational vision appeared (Ballard and Brown, 1982).

While there has never been a generally accepted paradigm in computer vision, there have been several unifying models that have extended beyond the laboratory of their originator. The labelling of line drawings in the blocks world has had a long history (Huffman, 1971; Clowes, 1971; Kanade, 1981) as a means of determining which surfaces belong to which distinct objects. The work of David Marr and his group on the primal sketch and its use in early vision (Marr, 1976 and 1982) has inspired much further research. The primal sketch is designed to be a rich, intermediate representation and is modelled on what takes place in biological vision systems--yet another paradigm.

In a pre-paradigmatic field, all problems seem equally interesting. In computational vision, some of the problems that have received attention include the detection of intensity discontinuities (edges) in an image (Shirai, 1975; Davis, 1975), detection of homogeneous regions in an image (Yakimovsky and Feldman, 1973; Kanade, 1980), the orientation of a surface ("shape from" shading: Horn, 1975; stereo: Grimson, 1981; contour: Witkin, 1981; texture: Kender, 1980; motion: Ullman, 1978), the three-dimensional representation of objects (Brooks, 1981), and the modelling of what might appear in scenes (Sloan and Bajcsy, 1977).

### 2.1.3. A New Focus

The major hypothesis of the dissertation concerns multiple information sources. As will be seen in later sections, other research has utilized multiple information sources, but only as a secondary facet of its solution to the problem at hand. The proposal is to make the notion of multiple information sources primary, and to relate the other aspects of problem solution to it.

In this sense the use of multiple information sources is being presented as a new paradigm. It is not presumed that this is a major paradigm that provides coherence for the whole field[2]. Rather, the paradigm generates a problem domain (i.e. how to best manipulate and coordinate the information sources) that can provide a focus for research.

The rest of this chapter will try to provide a convincing case for using multiple information sources as a focus of attention. Subsequently, the thesis will discuss the problems of dealing with information sources, both individually and in concert, in light of the the paradigm. One solution to those problems will be addressed in detail. Thus the overall (simplified) theoretical thrust of the thesis is as follows: major hypothesis--one useful solution to the problem of understanding

[2]It is clear that a crucial aspect of perception would be missing from this paradigm because it is possible, for people at least, to form a percept from a single information source such as a photograph.

images is to use multiple information sources (Chapter 2); secondary hypothesis--useful tools that allow one to take advantage of multiple information sources include schema-based languages, user interaction, and combined top-down and bottom-up control (Chapter 4). A particular implementation of those tools (Chapter 5) demonstrates their usefulness in the geographic domain of understanding aerial photographs.

## 2.2. Information Sources

### 2.2.1. What is an Information Source?

An information source is a set of input data. It can either be what is initially input to the system or an intermediate result that becomes available to later stages of processing. Standard information sources (henceforth sometimes abbreviated IS) in computational vision are digitized images (a two-dimensional array of intensity values bearing some relation to the amount of light reflected from objects in the scene in view) and line drawings (vertices and the lines between them forming some ideal representation of the edges of surfaces of objects in the scene). The IS is the input to a vision system and the output is some symbolic representation of the scene.

There is, of course, a range of symbolic representations that can be handled. A digitized image is at the low end of the scale. It is an iconic representation of the scene since there is a direct geometric mapping between them. A line drawing is more abstract because it only captures one aspect of the image, namely, the edges of surfaces. One can progress further up the scale to a representation of objects and finally to specific objects such as chairs and tables. The task of an image understanding system is to take an IS represented somewhere in that range and produce a "more abstract" output further up the scale.

Besides this range of how abstract a representation is, there exists an orthogonal dimension based on the type of information that can be found in an information source. This is illustrated clearly in the work done by Barrow and Tenenbaum on intrinsic images (Barrow and Tenenbaum, 1978). Their basic model consists of a stack of registered arrays which are iconic representations of the image. Each individual array contains a different kind of information: intensity, illumination, reflectance, orientation, and distance. In their system there is only one input, the intensity image; the other representations are built up from it through an iterative relaxation process. Thus the non-intensity arrays are both IS's and the final output representation. This is an example of how information sources can cooperate with each other.

Several of the other types of information used in computer vision include colour images (usually represented as three intensity images--for red, green, and blue), multispectral images[3], range data which gives distance directly, digital

[3]This is a generalization of colour images where certain bands in the electromagnetic spectrum are sampled. The most common examples of this are infra-red photography and satellite imagery such as Landsat which has 4 MSS bands.

terrain models (yielding the heights of points in the scene), sketch maps (giving the topological relationships between symbolic objects), and geographic data bases (containing the location and characteristics of landmarks).

## 2.2.2. The Conjecture

The goal of image understanding is to determine the mapping from what is actually sensed, the image, to the scene of interest. Unfortunately, the problem is vastly underconstrained in general. The factors resulting from properties of the objects, shape and surface material, cannot be separated from each other or from factors arising from illumination, shadows, viewing direction, path phenomena, or sensor noise (Woodham, 1981). Images are ambiguous and can be the result of an infinite number of scenes[4].

The hypothesis is that,

> Multiple information sources are useful in resolving ambiguities.

Or, simply,

> More is better.

This viewpoint is widely held by many vision researchers as will be seen in the discussion of systems that take advantage of

[4]This is the opposite of human vision in which images usually seem to be overconstrained and not ambiguous at all.

multiple IS's. There is a stronger conjecture that can be made, however, that has not been addressed before. This relates to the notion of different types of information.

> The more disparate that the information sources combined are, the better able they are to resolve ambiguities.

Or, restated,

> The more different, the better.

This is based on the intuition that similar aspects of the scene will be encoded in different ways by the different information sources. One thus has more chance of discovering what is important in the interpretation.

Some examples may clarify these ideas.

## 2.2.3. Illustrative Examples

If one starts with an intensity image, then the simplest additional information source to add would be another intensity image (more is better). To obtain something interesting, one would like to vary at least one parameter of the imaging process. Changing the viewpoint allows one to perform stereopsis which results in a depth map (Grimson, 1981; Baker and Binford, 1981). Changing the position of the source of illumination allows one to take advantage of a type of stereo called photometric stereo (Woodham, 1980a). If the time parameter is altered, one is able to determine shape from motion (Ullman, 1978).

A simple example will show how people use motion to good advantage. Consider the image of random dots in Figure 2.1. Clearly, there is nothing of interest in this image. Now consider Figure 2.2 below to be on a transparent overlay resting on top of Figure 2.1. The square would not stand out of the noise. However, if one moved the overlay around, the square would spring into view. Stop the motion and the square becomes indistinguishable from the background. If one had two intensity images separated in time, then a simple subtraction would reveal the square in this example (Anstis, 1970).

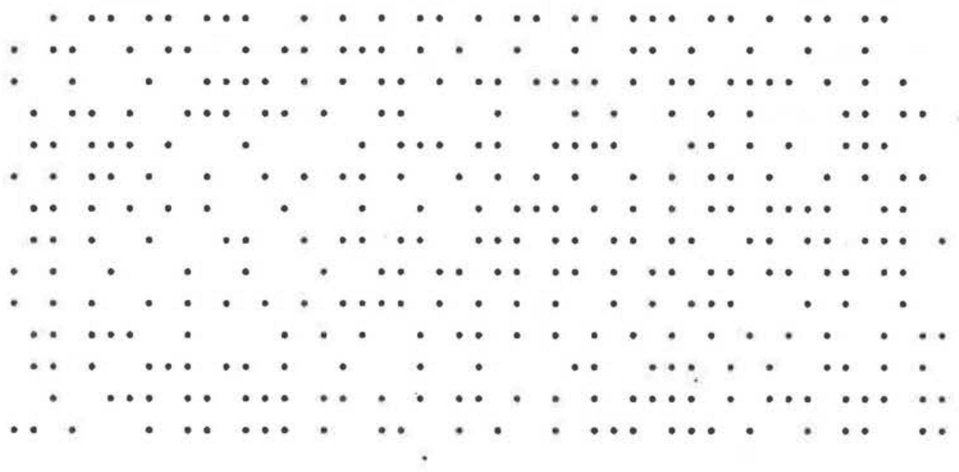


Figure 2.1 An Image of Noise

Figure 2.2 An Object of Interest

If the square had been a different colour, then it would have been apparent, to people, right away. In this case, one can imagine two intensity arrays, one for the colour of the noise, which is uninteresting, and one for the colour of the square, which is interesting.

In all of the above, the IS's have been the same--intensity images. Now consider the advantages of using different kinds of information. Figure 2.3 contains a line drawing that represents a letter drawn half way between an "A" and an "H". Now this image is deliberately ambiguous and no other similar image would be of any help. However, what if context was added as an information source? The expectations created in Figure 2.4a would probably cause us to conclude that the letter was an "H". Similarly, if the context was as provided in Figure 2.4b then the

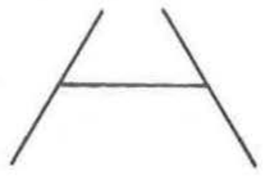


Figure 2.3 A letter

```
LETTER_BEFORE = T             LETTER_BEFORE = C
LETTER_AFTER  = E             LETTER_AFTER  = T

          a                             b
```

Figure 2.4 Some context

letter is likely an "A" (Neisser, 1967).

The Stroop effect is another example of how different types of information can influence perception (Neisser, 1976). Stroop had subjects read words which were the names of colours (e.g. "brown"). When the letters of a word were drawn in a different colour than the word represented (e.g. "brown" drawn in green ink), then subjects had more difficulty, and a significantly slower reaction time, than when the names and colours coincided. Here, then, is a case of where people are compelled to use multiple IS's even though it degrades their performance of a task.

## 2.3. Research that has Combined Information Sources

### 2.3.1. In Artificial Intelligence

The usefulness of combining a second source of information with a problem solver was recognized in the early days of AI research. Gelertner's work on proving geometry theorems was greatly aided by the information provided in a diagram which represents the problem[5] (Gelertner, 1963). When using a diagram to rule out impossible hypotheses, his program was able to prune 995 successors out of 1000, on the average, at each node in the search tree.

Information sources are directly related to knowledge sources as they were defined in the HEARSAY II speech under-

[5]Much in the same way that high school students are.

standing project (Erman et al., 1980). In HEARSAY each knowledge source is viewed as "an agent which embodies the knowledge of its area and which can take actions based on that knowledge" (Erman and Lesser, 1975, p. 483). This is similar to the ACTOR philosophy (Hewitt et al., 1973) of a community of experts working cooperatively to solve problems. However, whereas ACTORs seem best suited to an object-oriented solution (see Chapters 3.4 and 4), knowledge sources were implemented as production rules.

Information sources are related to knowledge sources in that each IS requires at least one KS to deal with it. Thus IS's and KS's can be seen to be the two sides of the same coin. The major difference between HEARSAY and this work is that the former concentrates on multiple KS's while the latter deals with multiple IS's. There is only one input IS in HEARSAY--the speech waveform. The other IS's exist as intermediate output that reside together on the blackboard and provide context for subsequent stages of processing, together with the relevant information retrieved from the Long Term Memory.

Despite that major philosophical difference, there are several methodological similarities between HEARSAY and the system described in Chapters 4 and 5. They both attempt to keep knowledge modular (in production rules on the one hand; schemata on the other). They both exploit a hierarchical representation (the blackboard; the. semantic network). Feedback control is used in both systems (hypothesize and test; a cycle of perception). They both strive to achieve graceful degradation when the information is poor.

Knowledge sources have been used extensively in AI systems, usually when it is useful to distinguish the types of knowledge that can be applied to the problem at hand. New knowledge sources can be integrated into DENDRAL because it is required that they encode their knowledge in the form of production rules which all communicate in terms of a common graph language. Data acquired from new types of instruments turned out to be very useful.

> . . . the impact of bringing just one additional source of knowledge to bear on a problem can be startling. In one difficult (but not unusually difficult) mass spectrum analysis problem, the program using its mass spectrometry knowledge alone would have generated an impossibly large set of plausible candidates (over 1.25 million!). Our engineering response to this was to add another source of data and knowledge, proton NMR. The addition of a simple interpretive theory of this NMR data, from which the program could infer a few additional constraints, reduced the set of plausible candidates to one, the right structure! This was not an isolated result but showed up dozens of times in subsequent analyses. (Feigenbaum, 1977, p. 1020)

Several researchers have argued for the use of disparate types of knowledge. Brown and Burton talk of the "synergism obtained by focusing the diverse capabilities of the procedural specialists" (Burton and Brown, 1975, p. 312). Davis has emphasized their importance (Davis, 1980a) and even raised their use to a principle:

> A fourth principle--exploiting redundancy--is nicely il-

lustrated by work on HEARSAY (Erman et al., 1980) that illustrated how redundancy can be a remedy for incomplete and inexact knowledge. The trick is to find multiple overlapping sources of knowledge with different areas of strength and different shortcomings. Properly used, the entire collection of knowledge sources can be a good deal more robust than any one of them taken alone. (Davis, 1982, pp. 6-7)

### 2.3.2. In Computational Vision

As in the larger field of AI, several researchers have argued for bringing "diverse sources of knowledge together" (Hanson and Riseman, 1978, p. 316; see also Flinchbaugh and Chandrasekaran, 1981, p. 391; and Witkin, 1981, p. 44). Russell has claimed that his system "must be able to work over a wide range of inputs from many sources" (Russell, 1979, p. 179). He does not explore the consequences of that belief.

Previous work that combines more than one intensity image to take advantage of stereo or motion has already been mentioned. Another classic use of several intensity images is found in pattern recognition where classification can be done by multidimensional cluster analysis (see Hall, 1979, Chapter 8). Each dimension represents a different intensity image which is derived by sampling from a distinct spectral band. Using different spectral bands to find distinguished features works because some properties stand out in certain bands more than in others. For example, the boundary between land and water is quite clear in imagery from band 7 of Landsat (near infra-red). This methodology works when the terrain is flat, but is not effective when the surface slope causes occlusions and shadows.

For more complicated scenes, digital terrain models can be combined effectively with intensity images (Horn and Bachman, 1978; Woodham, 1980b). A DTM records the elevation of all of the points in the scene. If the illumination direction is known, as in Landsat imagery, then it is possible to generate a synthetic image from the DTM, making some assumptions concerning the surface material of the scene objects plus some other imaging parameters. By registering the synthetic image to a real remotely sensed image, one can account for the effects of slope and a low illumination angle. At this point, the difference between the two images should ideally correspond to the characteristics of surface cover. In mountainous terrain this technique produces good results but work continues so that more aspects of the imaging process may be understood. That is, either more knowledge or more IS's are required.

Work has been done at SRI to combine a map data base with a digitized image (Tenenbaum et al., 1978; Bolles et al., 1979). As they explain,

> Map knowledge can provide important constraints on where to look in an image, what to look for, and how to interpret what is seen. Such constraints, properly exploited, permit the extraction of complex information without extensive computation. (Tenenbaum et al., 1978, p. 1)

The map data base contains three-dimensional locations and shapes of landmarks and monitoring stations. It can be indexed by location, entity name, and entity type. SRI researchers have produced a working system that demonstrates the feasibility of using this technique in several applications, such as measuring the volume of water in a reservoir, counting railway boxcars, counting ships in a harbour, and tracking roads. In contrast to the other systems examined, the map data base is a very abstract representation of information and very different in kind from an intensity image. This work clearly shows that it is possible to combine disparate types of information to good advantage.

A method of utilizing different road operators has also originated at SRI (Fischler et al., 1981). It uses one input IS, but many operators can be used to produce different internal images containing the likelihood that each pixel is part of a road. The global figures of merit are then optimized to produce the "best" road image. This program is discussed further in Section 3.1.2.

Disparate sources of information can be combined in systems that use evidential reasoning if the model knowledge has been suitably encoded (Wesley and Hanson, 1982; Garvey et al., 1981). This has been done in the VISIONS system using a dependency graph and a model of belief that manipulates both support and plausibility. It has the ability to effectively handle information that is imprecise, uncertain, and inexact.

There are other examples of model-based vision systems that will be described in Section 3.1. Where the model is explicitly defined, it can be considered an information source. In other cases the model knowledge is compiled into the procedures that carry out the interpretation. It is often difficult to draw a line between these cases.

## 2.4. Summary and Looking Ahead

This chapter has laid the framework for the thesis. The notion of using multiple information sources will strongly influence the descriptions in the next chapter outlining the current relevant pursuits in the fields of computer vision and knowledge representation. Moreover, this notion provides the rationale for the proposed solutions to the problem of how to best utilize information sources.

Information sources have been described as having three aspects. First, an IS can be input data that is fed to the system from the outside or it can be an intermediate result that is available to later processes or iterations. Second, IS's will vary over a wide range with respect to how abstract they are: from iconic representations all the way to "names" like "chair". Finally, in a dimension that can be orthogonal to the previous distinction, information sources can encode different types of information. In accordance with the two conjectures of Section 2.2.2 (more and different are better) emphasis in later chapters will be on using several input IS's containing different types of information and varying greatly in their symbolic nature.

Evidence for the efficacy of combining multiple IS's was presented in this chapter. Several advantages were demonstrated, and claimed, for their use. Primary is the ability to resolve ambiguities. Where information does not exist in one IS or is not conclusive it may be found, or be retrievable, in another. Furthermore, combining multiple IS's allows one to better deal with inexact and incomplete knowledge. If the "experts" that know about the IS's can cooperate, then together they can solve problems that as individuals they could not.

Besides sufficiency reasons for combining IS's (the problem can be solved with them), there are also efficiency reasons (the problem can be solved more quickly). For example, there are many "expensive" local operations to be carried out on intensity images, such as convolution for edge detection. If a second IS, such as a map data base or sketch map, can isolate "interesting" areas in the image, then these local operations need not be carried out over an entire image. Such a focus of attention mechanism can result in considerable savings in computation.

There are also potential disadvantages in trying to combine disparate types of knowledge. First, there is the correspondence problem. It is sometimes necessary to register representations of the image to each other, as in stereopsis, before any useful information can be derived. More generally, one must deal with the problem of conflicting information where one KS may derive a fact contradictory to what was discovered by a KS from a different IS.

The concepts of ambiguity, incompleteness, and inconsistency will be made more specific in the next chapter.

# CHAPTER 3

# A Framework of the Field

## 3.1. Model-Based Visual Perception

Computer vision systems can roughly be divided into two types. Domain independent vision (also known as image-based, low level, or early) is concerned with capturing as much information as possible from an image without recourse to knowledge of the underlying scene. Domain dependent vision (also known as model-based or high level) attempts to identify objects in the scene and make use of their semantics to aid the interpretation.

Domain independent vision applies to general, intrinsic properties of images (Zucker et al., 1975). Certain assumptions regarding the images are usually made to make the problems more tractable[1]. These systems generally deal with such image features as edges, surfaces, and reflectance.

This work is concerned with domain dependent or model-based vision. Model-based vision applies to scenes containing certain types of objects such as cubes, chairs, tables, roads, cars, trees, and houses. Only those objects can be "recognized". Domain dependent systems can take advantage of knowledge concerning the scene to influence the process of interpretation.

[1]Sometimes these assumptions relate to what can appear in the scene, making them model-dependent if not model-based.

For example, horses never have more than four legs. Nonetheless, model-based systems must contain some interface to an image-based component to provide the appropriate data from the image.

## 3.1.1. Different Types of Models

The first vision system to use simple models was developed by Roberts (Roberts, 1965). The "world" consisted of a cube, a rectangular wedge, a hexagonal prism, and combinations thereof. The three-dimensional model, encoded in terms of its vertices and lines, was matched to the two-dimensional image via heuristic search.

Further work in the blocks world domain allowed for more objects to be modelled (e.g. Falk, 1972, with 9 objects; see Mackworth, 1976, for a review). More complex three-dimensional objects can be modelled as generalized cylinders (Agin and Duda, 1973; Brooks et al., 1979) or generalized cones (Marr and Nishihara, 1978).

Having models of objects lets one take advantage of the known spatial relationships among them. Kelly's heuristic planning system (Kelly, 1971) used knowledge of faces to help locate the other prominent features (eyes, nose, mouth). Fischler and Elschlager extended the idea to include the relationships among all the features of the face (Fischler and Elschlager, 1973). In the ACRONYM system (Brooks et al., 1979; Brooks, 1981) a prediction graph is generated from the object models of such things as machine parts and aircraft. The constraints embodied in the graph restrict the possibilities when analyzing the image.

In Bolles's work on verification vision (Bolles, 1975), models of the objects (machine parts) and the scene enable the system to run almost totally top-down. When one has complete models of the objects, as in inspection tasks, the models can dictate what should be looked for.

Model knowledge can be captured in various ways. Freuder distinguishes between general knowledge about the domain and particular knowledge about the image (Freuder, 1976). His "active" knowledge modules were procedures embedded in a tree structure reflecting the control regime used for image interpretation of hammers.

The model can be represented explicitly in the form of tuples in a relational data base (Levine and Shaheen, 1981), in terms of frames (Sakai et al., 1976), in a semantic network (Bajcsy and Lieberman, 1974), or in production rules (Bajcsy and Joshi, 1978). The model knowledge can also be implicitly encoded in a procedural system (Shirai, 1975) or using fuzzy set theory (Jain and Haynes, 1982).

The input information source utilized by the above systems is generally an intensity image. Other IS's have been used such as line drawings (of houses: Mulder, 1979; of human body parts: Browse, 1982). Kuipers represented people's map knowledge and how they use it to move about in urban settings (Kuipers, 1977).

Two tasks are prevalent in systems dealing with the first stages of visual processing. Model knowledge has been applied to both edge detection (Shirai, 1975; Yachida et al., 1979) and segmentation--also known as region merging (Yakimovsky and Feldman, 1973; Tenenbaum, 1973; Tenenbaum and Barrow, 1976; Barrow and Tenenbaum, 1976; Hanson and Riseman, 1978a; Weymouth, 1981).

Few comprehensive systems have encompassed several aspects of vision. The VISIONS system, developed at the University of Massachusetts at Amherst (Hanson and Riseman, 1978a and 1978b; see Weymouth, 1981, for some more recent work) is a large, comprehensive system for understanding natural scenes composed of houses, trees, cars, and other objects. There is a single input IS (a colour intensity image) and a plethora of internal information sources. Two major repositories of knowledge are the short-term memory (particular knowledge) and the long-term memory (general knowledge), both of which are hierarchical and based on levels of abstraction. The levels go from vertices through segments, regions, surfaces, volumes, and objects to schemata.

The schemata are central to the VISIONS system. They are used to represent both objects and scenes. They provide information required to build up the volume (or surface) description of what is represented in an image. Finally, schemata guide the instantiation of hypotheses during interpretation. The system follows a hypothesize and test mode of control (focus-expand-verify) which, depending on the knowledge sources involved, could exhibit either bottom-up or top-down control.

The ACRONYM vision system has been developed at Stanford (Brooks et al., 1979; Brooks, 1981a and 1981b). Objects, modelled in three dimensions as generalized cones, and their relationships are represented in an "object" graph. The "restriction" graph contains constraints on the spatial relationships between objects. From these representations, a "prediction" graph is generated to hypothesize object to image feature matches. "Observation" and "interpretation" graphs are built during the analysis of an image.

The input IS is taken from the results of a line finder. From this, two-dimensional ribbons are found. Using a constraint manipulation system, these ribbons are matched against models in the restriction graph. As the interpretation graph is constructed, more specific models can be deduced (e.g. an L1011 is a type of wide-bodied jet).

An important aspect of these vision systems is the matter of program control. Most domain independent systems tend to operate in a bottom-up, linear fashion beginning with the image and working towards the possible objects in it. Domain dependent systems generally have a wider range of control strategies, but most contain either a top-down component or a feedback loop of some sort. Such systems have the problem of deciding which object to hypothesize (known as the chicken and egg problem: Mackworth, 1977b; or the parsing problem: Palmer, 1975), but once determined, these programs can make efficient use of object-specific heuristics to plan their strategies. *A priori* expectations derived from the models as well as the context resulting from previous interpretation both provide important clues (and cues) to guide the understanding of images.

## 3.1.2. Geographic Models

The domain of this dissertation concerns models of objects that might be seen in remotely sensed images. Objects one might try to distinguish in aerial or satellite imagery include roads, rivers, bridges, houses, mountains, and fields. Work has been done to study the cognitive aspects of how people acquire information from maps (Thorndyke, 1979) and several computational vision systems have been developed which use such knowledge.

Several projects at the University of Pennsylvania have utilized models of geographic entities. In one (Bajcsy and Tavakoli, 1973), models exist for water, land, rivers, lakes, bridges, and islands. Figure 3.1 shows the description of a river (Bajcsy and Tavakoli, 1973, p. 2A-57). This simple model contains three of the four major properties of a generic model: a list of properties which define the model (e.g. a river has texture); restrictions on those properties (e.g. texture is homogeneous); and relationships between objects (e.g. spatial relationship to bridge: below). The last aspect of a model is a

Rivers: Gray value of the water
Texture: homogeneous
Boundaries: open
Contrast: large
Spatial relationships to bridge: below
Topological relationships: continuous
Spatial relationships to land: surrounded by

Figure 3.1 A Model of a River

---

mechanism for determining if the features of a specific image match those in the model description. In this system these are derived procedurally. In a similar system, models of roads were used (Bajcsy and Tavakoli, 1976; see also Tavakoli and Rosenfeld, 1980).

Several other researchers have developed systems to find roads. Scientists at SRI have developed a method of using several types of information about roads.

> The approach is based on a new paradigm for combining local information from multiple and possibly incommensurate, sources, including various line and edge detection operators, map knowledge about the likely path of roads through an image, and generic knowledge about roads (e.g., connectivity, and width constraints). (Fischler et al., 1981, p. 201)

They divide all their information sources into two classes: Type I, where almost no false road elements are accepted, and Type II, where possible false elements are accepted but local, relevant parameters are accurately measured. The IS's are applied in different ways and all the results are optimized for maximum likelihood scores (using assumptions of continuity and width) to derive the "best" track. A requirement for the IS's is that the operator that produces them must also supply a numerical likelihood estimate for each point in the array. The multiple IS's are combined by reducing them all to a common representation plus ad hoc modifications in the optimization procedure.

Work was also done at SRI on map-guided interpretation. It was described in Section 2.3.2.

Selfridge uses "appearance" models for entities such as buildings, roads, and shadows (Selfridge and Sloan, 1981). A creditable feature of his system is the use of adaptive operators which allow parameters to vary when trying to find regions having certain areas, contrast, etc. (cf. Section 4.5.2 for a mechanism that accomplishes similar tasks.)

Nagao and his coworkers have described a vision system for interpreting aerial photographs modelled after HEARSAY-II (Nagao et al., 1978 and 1979). Common data is available on a blackboard and the knowledge is encoded in production rules. Object models for such items as crop fields, bare soil, grass land, roads, and rivers are not stored explicitly but are available procedurally as part of "model-driven subsystems". Interpretation proceeds from general properties (region segmentation) to more specific ones (selection of cue regions) and a feedback loop allows the context to aid in the classification of ambiguous regions.

Many maps use abstract symbols to represent objects, such as ■ for a building, ─┼─┼─┼─┼─┼─ for a railway. A line drawing containing only such stylized symbols is a sketch map. With only a little practice, one can form a cognitive map based only on the information found in a sketch map (see Figure 3.2).

Mackworth developed a program to interpret sketch maps called Mapsee (Mackworth, 1976c; see Ballard et al., 1978 for a different use of sketch maps in vision). Model knowledge is found in a table which is indexed by picture cues (features of line drawings such as TEE, OBTUSE L, and FREE END). The cues are ambiguous but network consistency (Mackworth, 1977a) is used

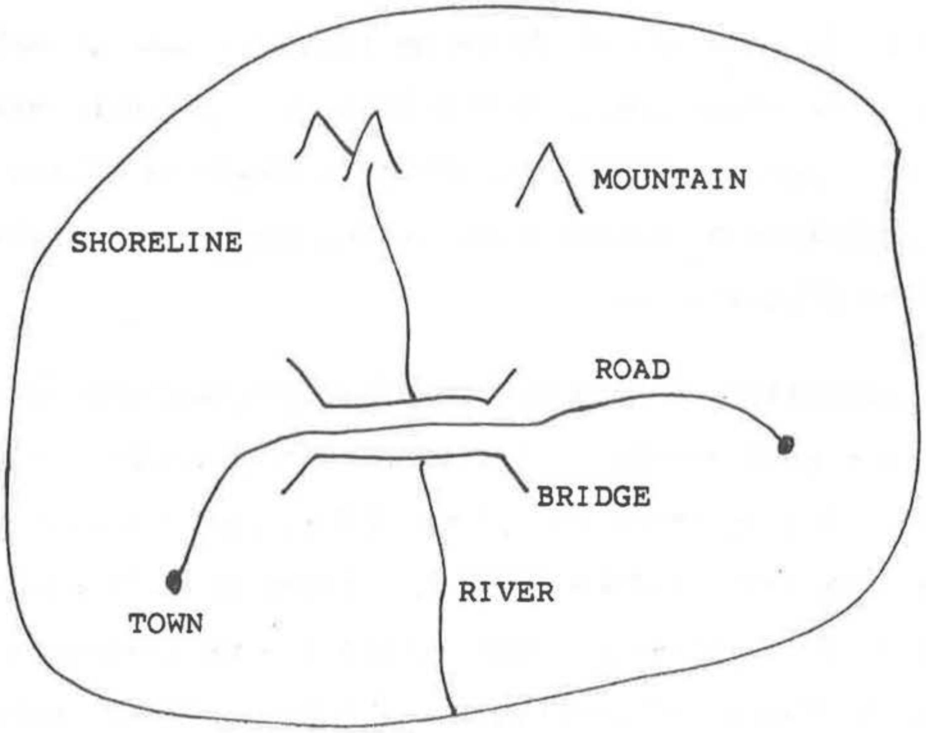


Figure 3.2 A Sketch Map

to reduce the label sets. This is done by taking advantage of the relationships between objects (e.g. rivers flow under bridges) that were used to build the table.

A second system to solve the same problem--Mapsee2--has also been developed (Havens and Mackworth, 1980; Mackworth and Havens, 1981). This system is schema-based and written in the language Maya (Havens, 1978). Schemata exist for all the elements of the domain from points through chains to roads, rivers, river systems, and so on up to "world". Associated with the schemata (which will be discussed in more detail in Section 3.3) are procedures for instantiating them plus a mechanism to carry on when that instantiation is completed. Another addition in Mapsee2 is the use of decomposition and specialization hierarchies to organize the schemata into a coherent network. The complete decomposition hierarchy is shown in Figure 3.3 and the specialization hierarchy that exists under geosystems can be found in Figure 3.4.

Recognition begins by creating hypothetical instances for entities that might correspond to each chain. These are restricted in some cases by the shapes of the chains. Thus, shorelines must be closed curves, towns must be blobs, mountains inverted V's. However, many shapes are ambiguous and can generate multiple instances. The newly created instances try to find a schema in the decomposition hierarchy to complete to but the instances must be consistent with specializations of the schema. For example, roads must be next to regions labelled as

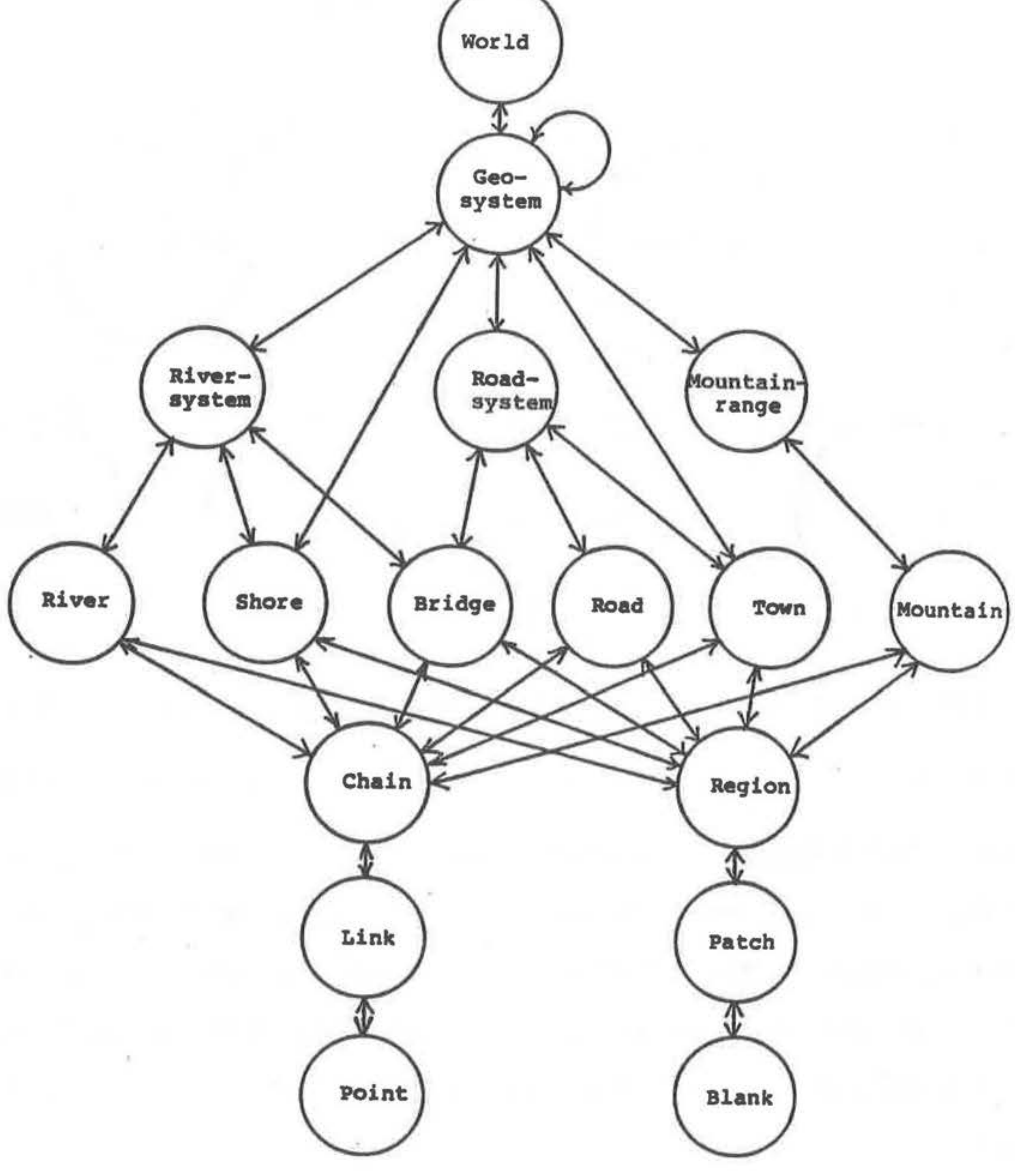


Figure 3.3 Mapsee2 Decomposition Hierarchy

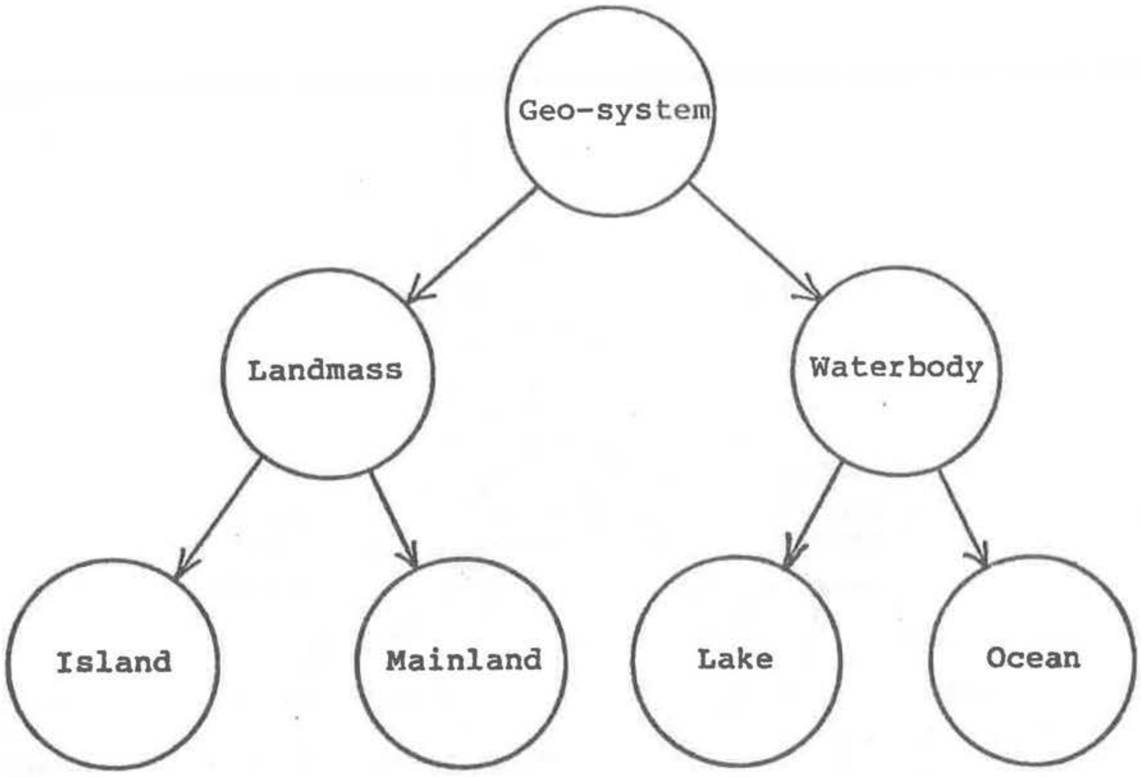


Figure 3.4 Mapsee2 Specialization Hierarchy for Geosystems

land. Inconsistent instances, such as a river going over a bridge, are pruned. Hypothetical, mutually exclusive instances form separate specializations. Unconnected instances cause the creation of new higher level instances, such as road systems. This continues for all the possible interpretations for each chain.

How might this work be extended? Mackworth states,

> . . . in the long run, . . . understanding [of LANDSAT images] would proceed more successfully if programs were able to accept advice, in the form of sketch maps, about

> the geography underlying the image. (Mackworth, 1978, p. 55)

This dissertation is concerned with such a system.

## 3.2. Possibilities in Matching Data to Models

Image interpretation can be considered a mapping between models and data. If the evaluation is top-down then the models form the domain and data elements the range. The situation is reversed for bottom-up evaluation. Besides the problem of incorrect interpretation, relating the wrong model with a datum, the relations may not always be one to one and onto.

Figure 3.5 illustrates the five possibilities that can occur when matching elements from the image (data) to models of scene objects. Example elements in an IS derived from an input image include edges (actually intensity discontinuities) and regions (where intensities are homogeneous). Some scene objects that might be modelled include roads, rivers, and bridges.

Ideally, one would like to discover a one to one relationship between a datum and a model. So, if rivers expect to be matched with a region from the image, it would be best if one particular region, say region-12, can be identified as a river (arbitrarily numbered, as river-2, to keep it distinguishable).

An inconsistent situation exists where more than one datum satisfies the requirements of a model. Alternatively, this is a situation where there is evidence both for and against an

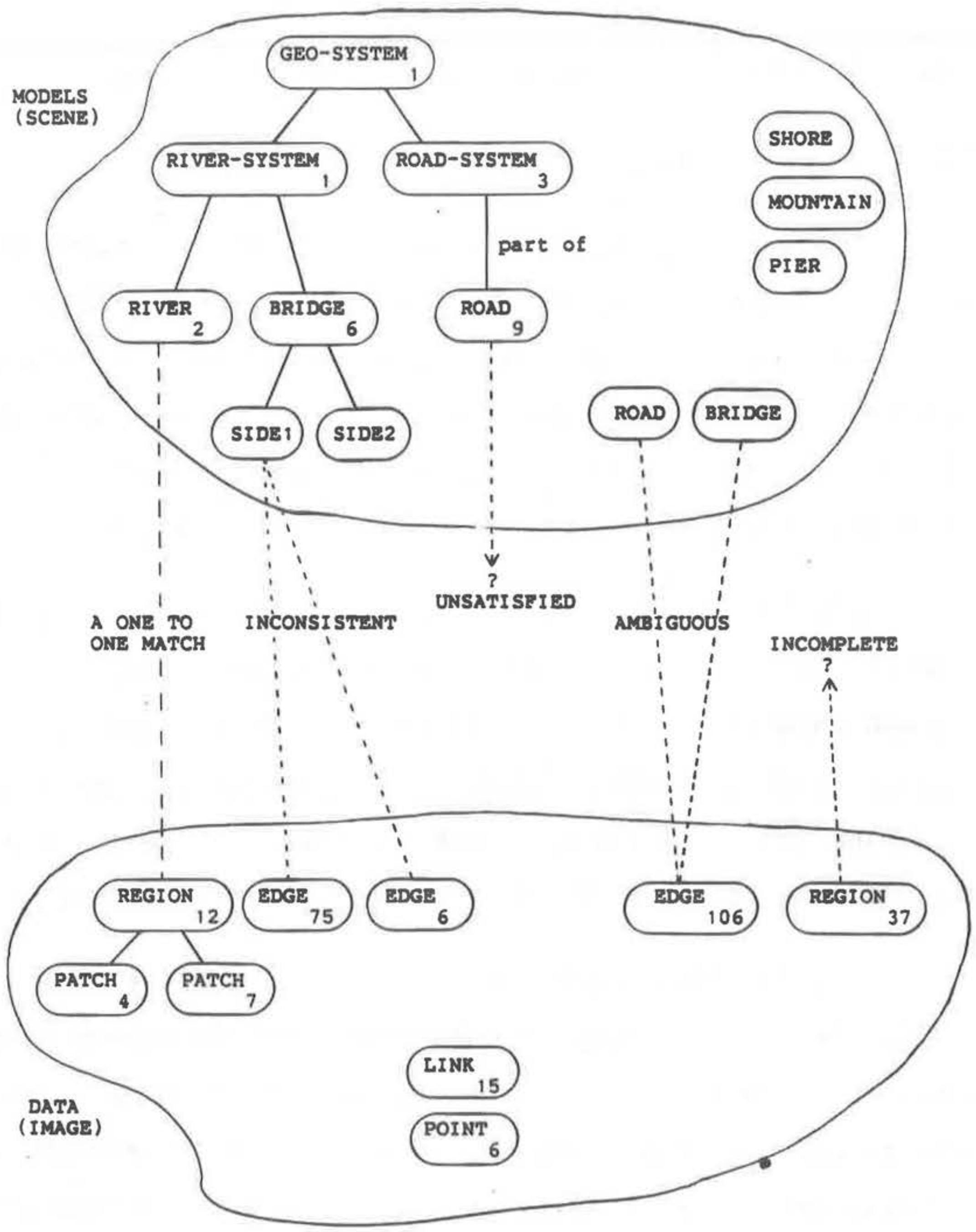


Figure 3.5 Possibilities in Matching Data to Models

interpretation. With regard to combining multiple information sources, the data elements need not come from the same IS. This can in fact cause great difficulty if the data from different IS's conflict rather than in agree.

On the other hand, if no data fulfill the requirements of a model, then the model is unsatisfied. Here multiple information sources can be an advantage, since there are more possibilities to find the data that will satisfy the model.

Ambiguity is generally the greatest problem facing computer vision systems. It exists whether the system is domain dependent or independent. Image features are ambiguous when they can be linked with more than one model. Model-based systems can take advantage of more global information such as the spatial organization of features so that single image elements are not the only facts available.

Finally, there can be data elements for which no mapping to a scene element can be found. This can occur for two reasons: either the "correct" model has not been represented in the system, or the model exists but is too rigidly defined to accept this "special case". Until vision systems include modules capable of abstraction and learning, this will remain a difficulty encountered in model-based vision.

One criterion for evaluating model-based vision systems is how well they handle the four problems described above. In particular, the problems of inconsistency and unsatisfiability are germane. These problems will be examined again with regard to the solutions that have been devised (Section 6.7.1).

## 3.3. Knowledge Representation (Descriptive Adequacy)

Knowledge representation schemes lie in a continuum from totally declarative to totally procedural. Declarative structures make knowledge explicit and available for modification. Procedural systems are easier to control and can make reasoning more direct. Schema-based representations are a compromise between the two extremes.

Schemata (Bartlett, 1932) or frames (Minsky, 1975) (or scripts, scenarios, units, templates, etc.) are a means by which associated facts can be related in a convenient way. Furthermore, they can encode the methods by which the facts may be utilized. Also, schemata that are related in an explicit way, in a semantic network, allow inferences to be drawn from the manner in which they are connected. Since they contain both a declarative part (the facts and the network links) and a procedural part (methods for manipulating facts), schemata can take advantage of the strengths of each.

There are many varieties of schemata that have been developed. Their merits can be judged on how well they encode the knowledge required to carry out the tasks for which they have been designed. Questions of descriptive adequacy include: how explicit are the facts? how uniformly are they encoded? can facts be modified as new information is discovered? how modular are individual schemata? how completely and correctly does the representation capture our intuitions? See (Barrow and Tenenbaum, 1975) for some other considerations.

Figure 3.6 shows a Maya (Havens, 1978) schema for a bridge in Mapsee2. This is an instance of a particular bridge. Maya distinguishes particular from general knowledge through the use of instances and objects respectively. The instance relationship is the only link between schemata that has a special meaning to the interpreter. Facts are encoded as attribute name/value pairs. Note that values can be S-expressions, pointers to other schemata (e.g. *CHAIN-5), or combinations of the two. Maya also supports tuplebases in which pattern-invoked, schema-specific procedures may be stored.

```
side1:       *CHAIN-3

side1-desc: ((57 . 33) (0.886914 . 0.461934) 54.1202)

side2:       *CHAIN-5

side2-desc: ((69 . 62) (-0.918062 . -0.396436) 47.918062)

regions:     (*REGION-1 *REGION-2 *REGION-3 *REGION-4 *REGION-5)

C/labels:    ((*CHAIN-3 . *BRIDGE) (*CHAIN-5 . *BRIDGE))

Q/models:    ((*RIVER-SYSTEM *RIVER-SYSTEM-1)
                  (*ROAD-SYSTEM *ROAD-SYSTEM-1))

instance->   *BRIDGE
```

Figure 3.6 A Maya Instance: *BRIDGE-1

A sample FRL frame (Roberts and Goldstein, 1977a and 1977b) is displayed in Figure 3.7 (from Rosenberg, 1977). FRL frames have more parts to an attribute, called a slot, than Maya does. Besides the name and value, there can be a default value and a REQUIRE predicate which restricts the values that may fill the

---

```
SUPPLY

   AKO             $VALUE       thing

   domain          $DEFAULT     carryover, production, imports

   old-carryover   $IF-ADDED    total-supply, carryover
                   $IF-REMOVED  total-supply, carryover
                   $DEFAULT     0

   production      $IF-ADDED    total-supply, carryover
                   $IF-REMOVED  total-supply, carryover
                   $DEFAULT     0

   imports         $IF-ADDED    total-supply, carryover
                   $IF-REMOVED  total-supply, carryover
                   $DEFAULT     0

   total-supply    $DEFAULT     total-supply

   use             $DEFAULT     domestic, exports

   domestic        $IF-ADDED    carryover, total-use
                   $IF-REMOVED  carryover, total-use
                   $DEFAULT     0

   exports         $IF-ADDED    carryover, total-use
                   $IF-REMOVED  carryover, total-use
                   $DEFAULT     0

   total-use       $DEFAULT     total-use

   carryover       $DEFAULT     carryover

   date            $DEFAULT     nowc
```

Figure 3.7 An FRL Frame:. SUPPLY

slot. There are also IF-ADDED and IF-REMOVED forms which are evaluated whenever a value is added or removed from the slot in question. These demons can be used to spread the effect of changes to the data base.

The one distinguished slot type in FRL is the AKO link (a kind of) and its inverse, INSTANCE. These links form a specialization or subset hierarchy which can be used for property inheritance. However, there are no particular restrictions on how the AKO links are to be used (see Brachman, 1982, for an explanation of how many different notions have been represented with AKO links). Also, there is no clear distinction between general or stereotypical knowledge and particular knowledge.

The UNIT package (Stefik, 1979; Smith and Friedland, 1980) also has slots as the basic attribute in units (frames). Slots are composed of names, values, defaults, datatypes (the type of the value such as integer or string; not to be confused with value restrictions that are like require clauses in FRL), and inheritance roles. As with FRL there is one inheritance hierarchy. However, the values that can be inherited are determined by their inheritance role. This provides another type of structure to the knowledge.

Besides properties, slots can have other definitional roles which in effect form other hierarchies. In particular, there is the part-of role, and its inverse, super-unit, which forms a decomposition hierarchy, and, a "relation" role which provides a general mechanism for describing binary relationships. Procedures can be attached to units, slots, and datatypes; messages can be passed between units or slots. These mechanisms have been used successfully in the building of expert systems (Stefik, 1980).

PSN is a representation language built from the basic entities of class and binary relations (Levesque and Mylopoulos, 1979). A class is a collection of objects which share common properties. A relation is a mapping between two classes. Employing simple basic operations on classes and relations (create/assert, destroy, fetch, and test), the PSN developers built IS-A (specialization) and PART-OF hierarchies--both of which can take part in the inheritance of properties. There is also the notion of a metaclass--a class whose instances are also classes--that can have properties not possessed by those instances. For example, the Student metaclass may have the property Average-Age that would not make sense in the Person instance.

Numerous other frame-based representation languages have been developed. Some knowledge representation languages like KRL (Bobrow and Winograd, 1977) have developed elaborate conceptualizations and imply a certain model of memory. Others, like Maya, can be viewed as extensions to Lisp that allow one to access knowledge more directly.

## 3.4. Knowledge Representation (Procedural Adequacy)

> . . . a data structure is not knowledge, any more than an encyclopedia is knowledge. We can say, metaphorically, that a book is a source of knowledge, but without a reader, the book is just ink on paper. Similarly, we often talk of the list-and-pointer data structures in an AI database as knowledge per se, when we really mean that they represent facts or rules when used by a certain program to behave in a knowledgeable way. (Barr and Feigenbaum, 1981, p. 143)

Schemata in a semantic network can be considered collectively as a data base which is manipulated by separate programs. Conversely, procedures attached to the schemata can control the process of interpretation. The latter case is an example of object-oriented programming.

Programming in an object-oriented language, of which Smalltalk (Robson and Goldberg, 1981) is a prime example, consists of defining objects and how they will respond to messages sent by other objects. Schemata are objects and most schema-based languages allow messages to be sent between schemata (e.g. Maya), slots within schemata (e.g. FRL), or both (e.g. UNITs and KRL). In an image understanding system, a message might come from a river schema, saying, "A river has been found in the image, it is river-6". The appropriate response might be that the river system schema checks if river-6 should be added to an existing river-system instance.

Questions of procedural adequacy relevant to object-oriented systems include: how effective are the control structures for effecting interpretations? how convenient is it to "program" with them? how well can procedures attached to different schemata cooperate? how schema-specific must the messages be?

Mapsee2 is a schema-based image understanding system (Havens and Mackworth, 1980; Mackworth and Havens, 1981) which is written in Maya (Havens, 1978). In Maya, one of the attributes a schema may have is a pointer to a database, called a tuplebase, of methods. Methods (functions) are retrieved and variables are bound to establish the current context by pattern matching. Hence each schema can have several local functions from which the appropriate one in the current context will be chosen. There is also a backtracking mechanism so that one can access the other functions if the first one does not succeed.

Mapsee2 currently interprets sketch maps in an almost strictly bottom-up fashion. Each chain (line) is a possibly ambiguous cue for several schemata such as roads, rivers, towns, bridge sides, or others. The appropriate schema-specific procedures examine the chain and determine whether it supports creating an instance, possibly hypothetical, to represent that entity. If so, that procedure is suspended and the functions associated with higher level objects, such as road systems, river systems, or geosystems, are initiated to build up the decomposition hierarchy.

## 3.5. Graceful Degradation (Robustness)

A computer program is said to be robust or display graceful degradation if its performance diminishes gradually when the conditions for which it was designed are not met. Marr defines the conditions for a robust program to be that "degrading data will not prevent one from delivering at least some of the answer" (Marr, 1976, p. 486). If an ungrammatical sentence is input to a natural language understanding program, then a robust system will still be able to analyze parts of the sentence[2]. A system that does not degrade gracefully might just go into an infinite loop because it "expected" a part of the sentence that was not there.

Graceful degradation is very important in interactive systems. Both the user and the system must have ways of recovering when communication is not complete. This has been called "graceful interaction" by workers on the Spice project at CMU (Hayes and Reddy, 1979; Hayes et al., 1981). They have identified several components of graceful interaction, including focus tracking (keeping track of what the conversation is about), identification from description (matching objects represented internally from external descriptions), an explanation facility (what the system is doing and why), and personalization (adjusting to the preferences of the user--cf. Section 4.3).

[2]People are very robust understanders of natural language.

Graceful degradation is also relevant to combining information sources. If an IS is removed, can the system utilize the remaining source(s) to advantage, or does it fail completely? As an extreme example, consider stereopsis and its two IS's. If one IS is removed, then it is impossible to calculate depth directly from a monocular image. Hence, in terms of its input IS's, stereo systems can not degrade gracefully when only one IS remains.

In systems combining different types of input information sources, more robust solutions can be developed. If one is using an "aid" IS to help interpret an intensity image (a map data base, DTM, sketch map) then a robust system will still be able to function with only the original image. One would not expect the results to be as good without the additional IS; however, correct partial results are almost always better than nothing at all. Thus, while robustness is a generally advantageous property for all computer programs, it can be particularly desirable in interactive systems and those attempting to combine information sources.

# CHAPTER 4

# How to Combine Multiple Information Sources

## 4.1. Introduction

Earlier in this thesis a case was begun for the usefulness of combining multiple information sources. Taking that paradigm as a starting point, there is still the question of how best to combine IS's so that they may be used in image interpretation. This chapter is concerned with one solution to that problem.

The major hypothesis of this dissertation is "Combining information sources is useful"; the minor hypothesis is that the methods discussed in this chapter are a way of realizing that goal. This chapter may also be thought of as describing tools which can be assembled (see Chapter 5) to form a vision system that can exploit different types of inputs.

The tools described below can be divided into three categories corresponding to the last sections of Chapter 3. Descriptive adequacy is the motivation for creating a new type of schema and a library of functions for schemata manipulation (Section 4.2). Information that comes directly from the user can be very useful. The means by which user information can be encoded is described (4.3).

Procedural adequacy is primarily the result of using a cycle of perception as the basic control regime (4.4). The cycle contains convenient points for interaction with the user (4.4.1) and can be used to generate top-down, bottom-up, or a mixed control strategy (4.5).

Several techniques have been devised to promote graceful degradation. Using context permits thresholds to be relaxed so that expected objects can be identified with weaker evidence (4.5.2). Island-driven control causes attention to be given to the most reliable hypotheses (4.5.3). Relating thresholds to global schemata connects model-specific knowledge with more relevant scene information (4.6). Finally, an inference mechanism based on clustering provides a flexible, incremental means of categorizing features derived from information sources (4.7).

Chapters 4 and 5 are closely linked. Chapter 4 introduces tools that can be applied to utilizing multiple information sources in perception. Chapter 5 describes three specific information sources and the actual implementation of a system which can analyze them using instances of those tools.

## 4.2. A Schemata Manipulation System

Few image understanding systems have used frames or schemata as a representation medium. Most of the schema-based languages described in Section 3.3 have been applied to tasks in natural language understanding and expert systems. From an AI perspective, one would hope that one representation scheme would suffice for all domains; however, there are differences in domains that can have an effect on how information can best be represented.

In particular, knowledge derived from images is very tentative and hypothetical, unlike the words input to an NLU program. This knowledge must be distinguished from generic knowledge and provision must be made to remove hypothetical instantiations that prove to be incorrect. Another difference is that, with visual knowledge, the decomposition hierarchy often turns out to be at least as important as specialization. Thus, a single distinguished relational structure between schemata is insufficient. Finally, the understanding of real imagery requires a great deal of procedural flexibility. Scenes can vary so much from image to image that uniform, syntactic methods have not usually met with much success.

Maya is a schema-based language that has been used in image understanding. Maya provides only a simple data structure to encode knowledge--attribute name/value pairs. The description that follows shows how Maya can be extended in ways that are sometimes similar to the other more complex knowledge representation languages, but often differ in accordance with the requirements of image understanding as described above. The result is a more differentiated data structure (Glicksman, 1982).

## 4.2.1. Schemata Have Four Parts

Schemata are made up of four distinct types of attributes. The VALUE type is used to store the facts that define the represented object (e.g. the "area" of river-6 is 76 pixels). It is an extension of Maya's attribute type to include FRL's slot facets and more. Although a value can be any Lisp S-expression, generally it will be a number, a list of numbers, or a string.

VALUEd types provide a slot that can be filled with a value. As in FRL, the slot can have expressions associated with it, such as a default or the required properties of the slot filler. Furthermore, functions can be initiated when the value is added, removed, or needed. Unlike FRL, a function also exists which is invoked when the value is modified. This flexibility is used to maintain the consistency of the interpretation and to spread the effects of changes to the slots.

The final modifier that can be associated with a VALUEd type is confidence. This allows one to associate a reliability factor with the components of a schema that can be used to calculate the overall CONFIDENCE of the schema. It is useful in image understanding where the existence of objects is not an all or nothing proposition (see Davis and Rosenfeld, 1981, for a review of relaxation methods). Confidence is also related to the issue of graceful degradation since the system should have less confidence in its results when presented with poor data (Jain and Haynes, 1982). Figure 4.1 contains an example of a schema having these attribute types.

```
length       VALUE:        46            DEFAULT:     nil
             REQUIRED:     (greaterp %val 0)
             IF-ADDED:     (and (sgetv %name 'width 'n)
                                (sputv %name 'area
                                       (times %val width)))
             IF-MODIFIED: (and (sgetv %name 'width 'n)
                                (sputv %name 'area
                                       (times %val width)))
             IF-REMOVED:   (sputv %name 'area nil)
             IF-NEEDED:    (and (sgetv %name 'width)
                                (sgetv %name 'area)
                                (quotient area width))
             CONFIDENCE:   37

width        VALUE:        6             DEFAULT:     nil
             REQUIRED:     (greaterp %val 0)
             IF-ADDED:     . . . similar to length . . .
             CONFIDENCE:   48

area         VALUE:        276           DEFAULT:     nil
             REQUIRED:     (greaterp %val 0)
             IF-NEEDED:    (and (sgetv %name 'length)
                                (sgetv %name 'width)
                                (times length width))
             CONFIDENCE:   42

intensity    VALUE:        102           DEFAULT:     nil
             REQUIRED:     (and (greaterp %val -1)
                                (lessp %val 257))
             IF-ADDED:     (modify-interpretation-range %val)
             IF-MODIFIED: (modify-interpretation-range %val)
             IF-REMOVED:   (modify-interpretation-range %val)
             IF-NEEDED:    (quotient (sum-all-pixel-values %name)
                                     (sgetv %name 'area))
             CONFIDENCE:   26

aio->         %river       apo->         %river-system-2

ako->         (%transportation-system-1 %waterbody-3)

neighbours-> (%bridge-7 %river-8)

CONFIDENCE    43

CONF-ALG      (cluster (sgetv %name 'intensity) riv-intensities)

PROCEDURES:   BOTTOM-UP—> river-bottom-up

              TOP-DOWN—>  river-top-down
```

Figure 4.1 A Schema: %river-6

The schemata are contained in a semantic network (or knowledge base), with relationships indicated by LINKs--the second distinguished part of a schema. Any number of binary relations can be formed, and the system maintains the pointers and their inverses and allows for several forms of automatic traversal of the network. LINKs can be used with VALUEd types to generate property inheritance, usually down a specialization hierarchy. Knowledge sharing of this kind can often compensate for incomplete data. Some relations induce hierarchies which create interesting possibilities for control strategies (see Rosenthal and Bajcsy, 1978).

One prominent built-in relation is INSTANCE/AIO[1] which as in Maya is used to separate general or stereotypic objects from their realizations in a scene. Instances can be used to explore several possible interpretations without any commitment to their ultimate existence. Other relations that would be generally used are SPECIALIZES-TO/AKO to form a specialization or subset hierarchy and DECOMPOSES-TO/APO to form a decomposition or sub-part hierarchy (see Figure 4.1).

The CONFIDENCE attribute is a number representing the overall confidence one might have in a schema at a given time considering the available information. The CONFIDENCE value is available to a global scheduler and can be used to encourage

[1]Links are referred to as link/inverse. AIO stands for An Instance Of; AKO for A Kind Of; and, APO for A Part Of.

evaluation of the most promising interpretations.

Associated with this attribute is a schema-specific algorithm to modify the confidence value whenever new VALUEs are found for this object or its components. Many "belief systems" rely on uniform functions for calculating reliability. For example, MYCIN uses the minimum certainty factor of its antecedents (Shortliffe and Buchanan, 1975; see Goebel, 1977, for a review of other systems which use the maximum or average values). Using an algorithm provides the flexibility to consider the difficulty and importance of completing the interpretation of an object as well as the probability of its existence. It allows for schema-specific knowledge to be used to weight the importance of the VALUEd types as well as components and specializations of a schema when calculating confidence.

The fourth distinguished part of a schema is its procedural attachments. Besides the procedures that can be attached to attributes (VALUEd and CERTAINTY types), associated with a schema there are generally procedures which are used to instantiate them (i.e. fill in their slots) and control the interpretation. These would include procedures to invoke the schema as a model (top-down) or to account for data (bottom-up). The procedures would also typically send out messages which would cause other schemata to become "current" and other procedures to be evaluated.

### 4.2.2. Manipulating Schemata

Numerous functions are available to manipulate schemata so that the information they encode can be utilized. Appendix A describes the functions available in the MAIDS system (Section 5.2) which is a particular implementation of the representation scheme described in the last section.

Essential functions include those for creating schemata (screate--for stereotype objects; snewi--for hypothetical instances) and destroying them (seraseo and serasei), plus one for merging the attributes of two instances into one (smerge). Also there are basic functions for modifying and accessing the knowledge base. Table 4.1 lists the functions available for each of the attribute types.

In addition to these standard functions, there are several more that add to the usefulness of the schemata. Sgetv can be used to fetch defaults and/or values along a specified LINK which can be used for property inheritance. One can determine the shortest path between two schemata along a certain link

| | ADD | MODIFY | REMOVE | FETCH |
|---|---|---|---|---|
| any attribute | sputa | sputa | | sgeta |
| VALUEd | sputv | sputv | sremovev | sgetv |
| LINKs | saddl | saddl | sremovel | sgetl |
| CONFIDENCE | sputc<br>sputcl | sputc | | sgetc |

Table 4.1 Modification and Retrieval for Schemata Attributes

(slink?), or discover whether a path exists between two schemata (sanylink?). The semantic network can be made consistent by adding missing back pointers and removing superfluous LINKs (sconsist).

Given a type, such as LINK, it is possible to determine all the attributes in a schema that are of that type (sattr). Conversely, if one knows the name of an attribute, then its type can be found (sattrtype). Also, several pretty printing functions exist to show the components of schemata (sprint) or the branches of the semantic network (sprintn and ssprintn).

The data structures for schemata representation plus the manipulation functions provide building blocks for a vision system that can combine multiple IS's. Their greatest assets are modularity, the flexibility to encapsulate diverse interpretation strategies, and the interrelationships among schemata which enables them to cooperate.

## 4.3. Interaction as a Source of Information

Most computer vision systems have not attempted to use advice from a human user. Whereas totally automated systems are the ultimate goal in this field, it is often the case that minimal human intervention may provide the crucial information that will enable interpretation to succeed. Furthermore, if a truly symbiotic relationship can be formed between man and machine then it would be possible for more difficult problems to be solved through cooperation than could be handled by man or machine alone.

Even though the subject has not been dealt with in great detail, several researchers have advocated the usefulness of human interaction in both vision systems (Barrow and Tenenbaum, 1975; Agin and Duda, 1975; Barrow and Tenenbaum, 1976; Stockman, 1978) and expert systems (Nii et al., 1982). One of the reasons was well described as follows:

> The substantial amount of ad hoc world knowledge required to plan perceptual strategies is most reasonably acquired in an incremental fashion. The system should thus be designed to request additional information from a user at times of failure, indecision, or on encountering a new object and to incorporate this information immediately in a revised strategy. (Tenenbaum, 1973, pp. 8-9)

Human interaction would be most useful in current vision systems to resolve problems due to ambiguity or indecision. If a vision program has reduced the number of possibilities for an interpretation down to a small number, then very little "information" is required from the user to resolve the difficulty. Furthermore, interaction can be useful in combating the other problems of model-based vision (as outlined in Section 3.2): inconsistency, unsatisfiability, and incompleteness.

One of the criteria for the success of a computer vision system is that it display graceful degradation (Section 3.5). Hence, if the user chooses not to participate in the interpretation process, the system should still be robust enough to continue without the additional guidance. The more useful information the user chooses to provide, the better the results one would expect. If the user wishes to input information, it would be useful for a vision system to accept it.

An interactive vision system should only be making requests of the user at times when critical decisions must be made. However, how is the system to know what the user considers critical? One way is for the system to have a user model that has knowledge of the preferences and habits of the individuals that use the system (Rich, 1979a and 1979b). Then the system can refer to the model in the context of the current decision when determining if the user should be queried and how.

## 4.3.1. Accepting and Accomodating Information from Users

Man-machine communication is best achieved when the computer is able to relate in a natural manner. Thus natural language is to be preferred over obscure programming languages. In image understanding where pictorial information is so important, graphic input and output becomes critical. Graphic input usually consists of either pointing at parts of an image or drawing.

Sketch maps can be considered a type of interaction between the user and the vision system. They depict certain standard symbols which are abstractions of what might appear in a scene (cf. Section 3.1.2). They contain information concerning what is in a scene as well as the topological relationships among the entities.

More direct information might be input to a vision system via text. Whereas a sketch map retains some analogic properties of the intensity image, direct input is generally not iconic. It is also usually much more abstract than an intensity image. Examples include specifying the existence of an object, its location, or its characteristics.

To be able to accomodate disparate types of knowledge requires a method of representation independent of the nature of the images. The schemata described in the last section (4.2) provide a flexible, modular representation scheme that is able to store and utilize advice from a user. Attached procedures can provide the interface between man and machine so that slots can be filled with values directly. These values can then become part of the context which influences subsequent processing.

In interactive systems there is also the question of what information to provide to a user, when it is appropriate to display it, and, conversely, what and when information should be received. The next section on a cycle of perception indicates three places in the feedback loop where interaction can occur.

## 4.4. A Cycle of Perception

Cognitive psychologists have examined the role of schemata as collections of structures and processes that both accept perceptual information and direct movements and exploratory activities (Neisser, 1976). Neisser describes how they might be used as follows:

> In my view, the cognitive structures crucial for vision are the anticipatory schemata that prepare the perceiver to accept certain kinds of information rather than others and thus control the activity of looking. Because we can see only what we know how to look for, it is these schemata (together with the information actually available) that determine what will be perceived. Perception is indeed a constructive process . . . At each moment the perceiver is constructing anticipations of certain kinds of information, that enable him to accept it as it becomes available. Often he must actively explore . . . These explorations are directed by the anticipatory schemata, which are plans for perceptual action as well as readiness for particular kinds of optical structure. The outcome of the explorations--the information picked up--modifies the original schema. Thus modified it directs further exploration and becomes ready for more information. (Neisser, 1976, pp. 20-21)

Neisser calls this continual process the perceptual cycle. A slightly modified version can be seen in Figure 4.2.

A similar feedback loop has been proposed specifically for model-based computer vision (Mackworth, 1978). Its four stages are cue discovery (object), model invocation (schema invocation), model verification (schema instantiation), and model elaboration (exploration)[2]. Mackworth's "paradigm" is called the cycle of perception.

Depending on where the cycle is entered, one will observe different modes of operation which correspond to the traditional methods of control in computer science. If the cycle begins at

[2]What has been labelled "schema invocation and instantiation" in Figure 4.2 is simply called "schema" in Neisser's perceptual cycle.

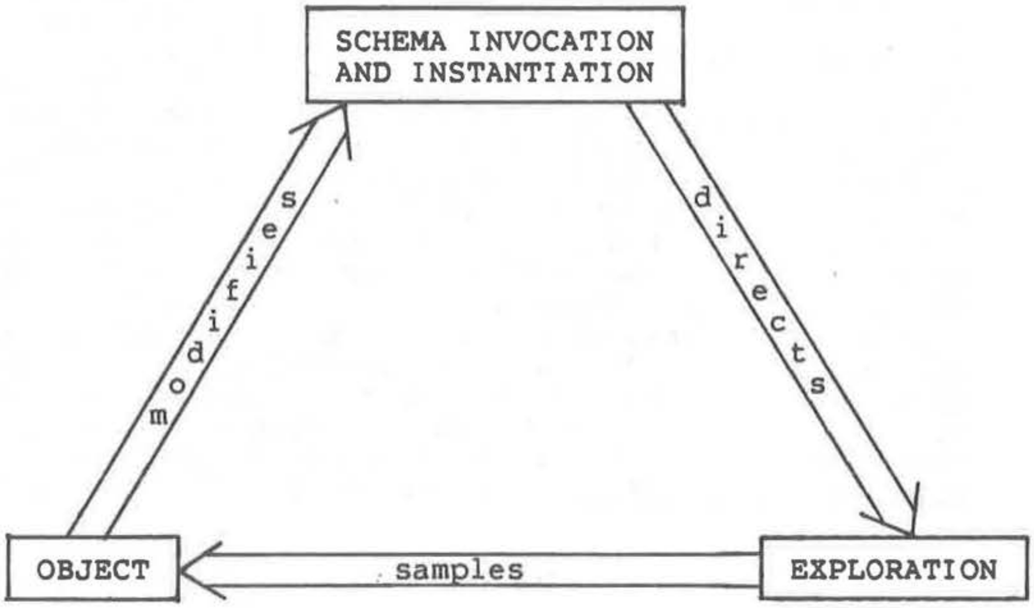


Figure 4.2 A Cycle of Perception

the "object" stage then bottom-up, or a linear stage model, can result. Cues from the image derived from objects in the scene cause schemata (models) to be invoked and instantiated. In a linear stage model, processing would stop at this point: the invoked model is a classification. However, an iterative process would use the information just obtained to direct further exploration in the image and the cycle would continue.

If the cycle is entered at the stage of schema invocation, then top-down behaviour would result. The model knowledge contained in the schema will be used to hypothesize the existence of objects in the image. There will also be information concerning how and where to obtain the information in the image that will verify the hypotheses. So, the exploration module will cause the image to be sampled to find the appropriate information. Depending on what is found in the image, the schema will be suitably modified. The instantiated schema can then request more information to confirm its existence or it can use the knowledge gained to help hypothesize the whereabouts of other objects.

In this way, the cycle of perception promotes cooperation among the schemata. As objects are instantiated they communicate to other schemata to give advice and/or to build the semantic network. This in turn initiates another loop around the cycle as a new schema is instantiated. The schemata work as individuals (via attached procedures) and together (via messages) to bring about the interpretation of particular information sources.

The cycle of perception is similar to other feedback paradigms such as hypothesize and test (e.g. Erman and Lesser, 1975). It is particularly well suited to a schema-based system because it identifies the natural places where schemata can move into and out of the focus of attention. The consequences of this are discussed in the following section (4.5).

### 4.4.1. Interaction in the Cycle of Perception

The cycle of perception provides convenient points for communication with the user. At each node in the cycle there is a particular, useful kind of information that can be exchanged. These inputs and outputs are shown in Figure 4.3 and are described below.

OUTPUTS

GLOBAL: When a schema is invoked, the model type (e.g. river)

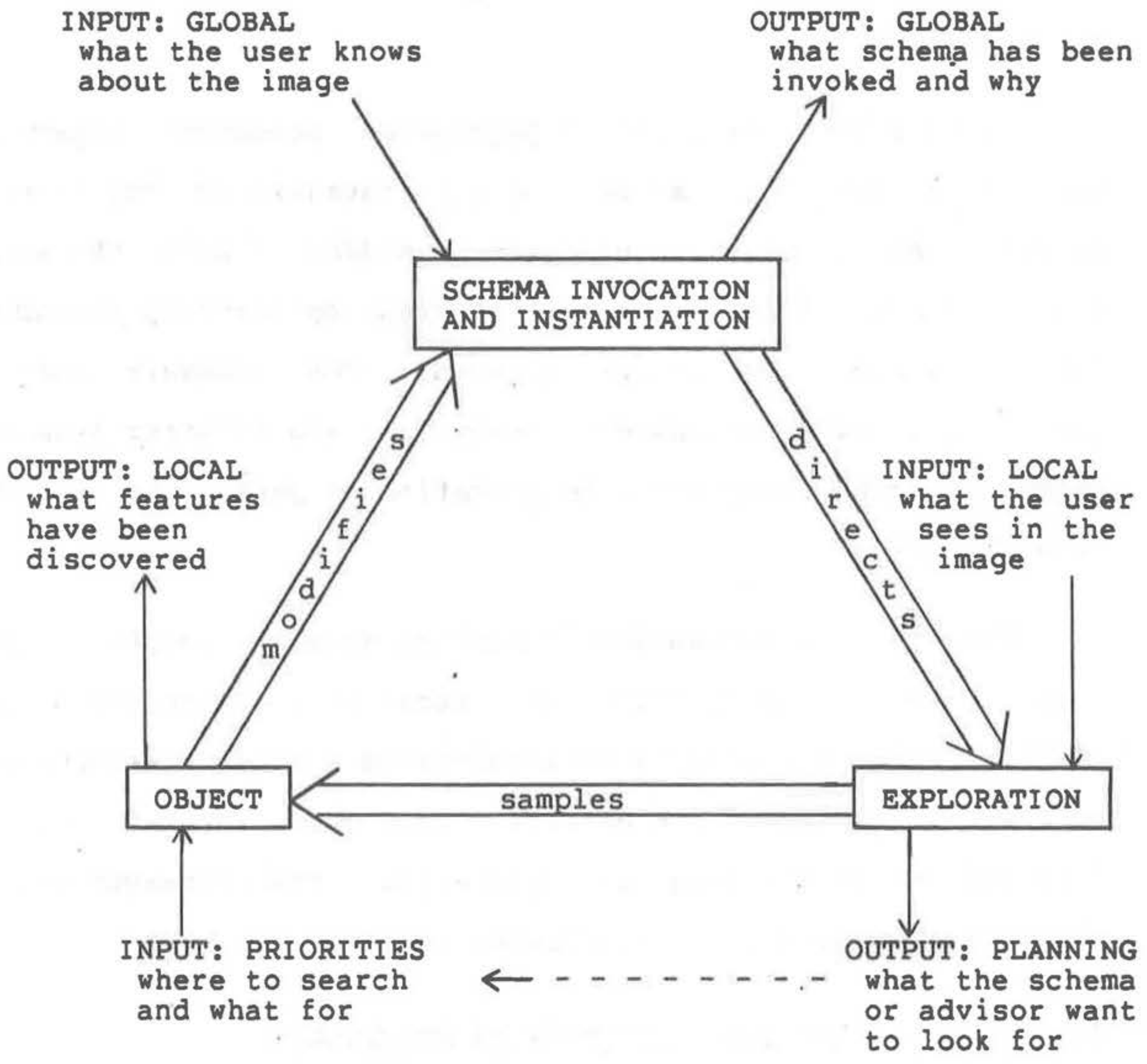

INPUT: GLOBAL
what the user knows about the image
OUTPUT: GLOBAL
what schema has been invoked and why
SCHEMA INVOCATION AND INSTANTIATION
modifies
directs
OUTPUT: LOCAL
what features have been discovered
INPUT: LOCAL
what the user sees in the image
OBJECT
samples
EXPLORATION
INPUT: PRIORITIES
where to search and what for
OUTPUT: PLANNING
what the schema or advisor want to look for

Figure 4.3 Interaction in a Cycle of Perception

can be described for the user so that he/she knows what the current focus of attention is. A curious user would often like to know the reasons behind that choice. Furthermore, as the schema is instantiated--by having its slots filled with values--the information can be output (e.g. the "area" of the river is 276 pixels).

LOCAL: The information derived from the image can be output when the image is sampled. It can be in the form of verifications of hypotheses directed by the schema (e.g. rivers have low intensity values: the average intensity of this region is 102: so succeed), facts concerning the image features that may be relevant to the schema (e.g. the long axis of the region is 46 units), or data that can cause a shift in attention (e.g. one of the neighbouring regions has a high average intensity; it can be a bridge that the river is flowing under).

PLANNING: This is a list of possibilities for further processing. As they are executing, the schema-specific procedures can make inferences about other models that should be invoked. They will then send a message to alert the appropriate schema. Those messages plus the user's input make up the possibilities list.

The schema-specific procedures will give up control for one of three reasons. The procedure might succeed and the current schema will become part of the semantic network. It might fail, in which case the hypothetical schema will be destroyed. Or it might suspend to wait for more information.

Attention would then normally shift to the first entry in the possibilities list.

The possibilities list is a priority queue. Its entries are ranked by the schema that sent the message. The value given by the schema will generally be a factor of its own confidence (cf. Section 4.2.1) and the certainty of the inference that caused the message to be sent. The rankings are shown to the user so that he/she can see how processing will continue and can assert his/her priorities by rearranging the queue which is done via priorities input.

INPUTS

GLOBAL: The user can enter general or specific information about the scene. General data includes parameters that affect all levels of processing, for example, the scale of the aerial photograph or the location of the sun. Specific information pertains to specific objects. For instance, one can indicate that there is a road in the picture. This information can cause entries to be added to the possibilities list or might cause global schemata or variables to be modified (see Section 4.6).

LOCAL: More specific information can also be introduced concerning the interpretation that is taking place. This might take the form of an interaction with a specific routine. For example, System: "Is the bank of the river edge 12 (displayed in red) or edge 26 (blue)?" The user would point at his/her choice. Or the user might just input what he/she sees in the image: region 72 is a river. In the former case, the data would be used to influence how a schema was instantiated; in the latter, an entry on the possibilities list may be added or modified.

PRIORITIES: The user can influence subsequent processing by modifying the priority queue. Rearranging the entries changes the search and instantiation priorities of objects. As an example, if one is interested in bridges, move all the entries relating to bridges to the head of the queue. Objects that are of no interest or that are perceived to be fallacious can be removed from the queue. By altering the parameters within the entries, one can modify the interpretation context. This would include changing the location in the image where the object will be sought.

The cycle of perception is well-suited to a schema-based, object-oriented vision system. It allows feedback from previously instantiated schemata to provide a more informed context for subsequent processing. Also, interaction is facilitated by having clearly distinguished points in the cycle where different types of information can be exchanged. The next section describes a complex control strategy which, because of the cycle of perception, can take advantage of both general and particular knowledge.

## 4.5. Mixing Top-Down and Bottom-Up Control

Besides the cycle of perception, control is strongly influenced by the hierarchical relationships between schemata. In an object-oriented system, the connections between the objects determine which pairs of schemata can most profitably communicate. Messages can be sent from any schema to another; however, part of the goal of interpretation is to join all of the schemata into a unified network necessitating significant communication between neighbouring nodes.

The two important hierarchies in terms of control are specialization and decomposition. An example where these have been intertwined is found in Figure 4.4 (modified from a classification in Lillesand and Kiefer, 1979, and the decomposition and specialization hierarchies in Mapsee2). This represents only general or stereotypic knowledge. A separate but related instance version of this graph would be created during the interpretation of an image.

### 4.5.1. Top-Down and Bottom-Up Control

In a graphic form, the standard control paradigms are apparent. Top-down, model-driven behaviour will be observed when schema-specific routines respond to messages from "above" in the hierarchies. Bottom-up, data-driven control comes from below, since the objects at the "bottom" of the graph (runway, road, river) can be considered to be the most primitive, "closest" to the data. The data (the intensity image and the

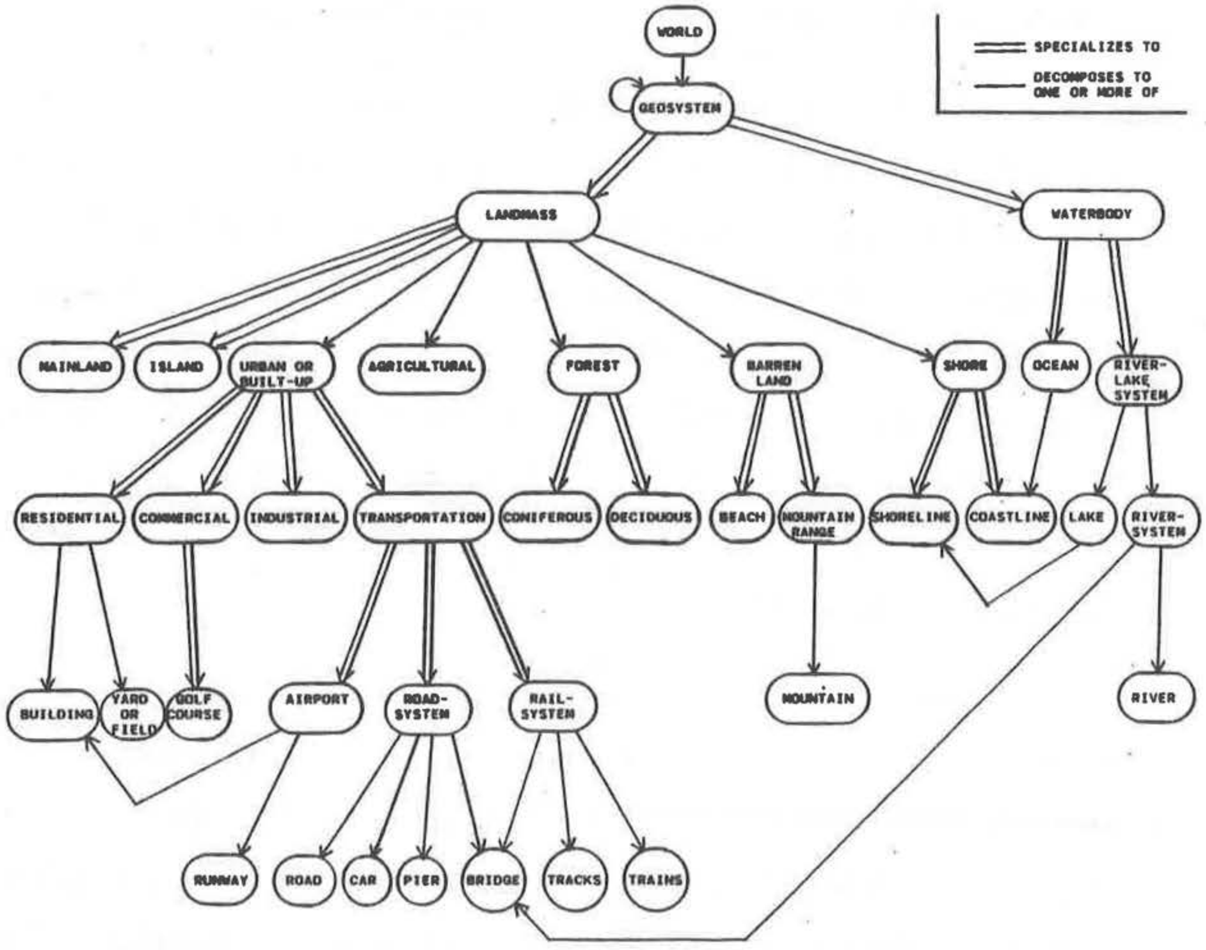


Figure 4.4 Hierarchies in a Geographic Domain

sketch map but not necessarily that coming from the user) can be thought of as being another layer "below" the depicted schemata. The relationship is not specialization or decomposition but some type of mapping. The directions in the graph make sense because the graph has been displayed with the schemata ordered from the bottom by increasing generality and composition.

Top-down control can take two forms. The first is like top-down parsing: e.g. if one wants to establish a geosystem then find either a landmass or a waterbody (G -> L | W). Model knowledge is used at each stage in deciding how to move down the hierarchies until a bottom level schema (terminal symbol) is reached. If it can be instantiated from the IS's then the hierarchies that have been built up can remain in place. Otherwise, control must back up to a choice point and a different branch must be taken.

The other type of top-down control can take effect only after part of the interpretation has taken place. Previous results help form the context of the current stage of processing. The context plus the model knowledge of the schema are used to produce a more efficient search. For example, if it has already been established that a bridge exists, then if control currently resides in the road system schema, a good strategy would be to look for a road going over the bridge. Since the location and orientation of the bridge are known, this greatly constrains the resulting search for evidence of a road.

## 4.5.2. Relaxing Thresholds

This second type of top-down, or expectation-driven, control can also be used to relax thresholds. Thresholds are those cutoff points that schema-specific routines must use to determine whether the features in an IS are sufficient to instantiate the object. For example, the intensities of pixels representing water are usually less than those corresponding to land. A threshold for water is an intensity value that classifies as water all pixels having lower intensities. However, it is generally true that if one tries to find an intensity value that divides all pixels into two categories (water and land), there will always be some pixels that are classified incorrectly.

The situation is not desperate if one considers a single classification, such as water. It is often the case that a restricted range of intensity values can be found such that any pixel having an intensity in the range can reliably be classified as water[3]. While this implies that non-water objects will not have intensities in that range, there will be pixels corresponding to water that are misclassified. But if the schema-specific routine for a waterbody is executing, perhaps at the lake node, in a context where water is expected to be found, then the threshold can be relaxed (i.e. the range extended) so

[3]This will not always be the case in complicated images containing shadows, clouds, light sources, etc. but then, the reliability would be low. Also, single pixel values are prone to several types of errors, so that groups of pixels are preferable--cf. Section 5.3.1.2.

that more of the image can be correctly classified.

In this way, one starts from a conservative position that will only accept feature values that are reliable indicators of the presence of an object (the principle of least commitment--Marr, 1976). As the expectations rise for existence of the object, the thresholds can be relaxed so that less reliable feature values will still verify its presence. The current context plus model knowledge determine how much the thresholds should be relaxed.

Returning to the example, if the range of possible pixel intensities is [0,255] and the classifications for pixels include water, road, field, urban, and mountain, then there might be a range [20,50] such that all pixels having values in this range are water. That is, no pixels to be classified as road will have an intensity values in this range. Another range [0,75] would include all pixels to be classified as water but might include other categories as well. Thus if there is no *a priori* knowledge of the pixels being considered, use 20 and 50 as decision boundaries. If there is a high expectation that a pixel represents water then test whether it is between 0 and 75. Intermediate expectations can prompt the use of values somewhere between these extremes, such as 10 and 62.

More formally, if C is the set of all possible classifications (C0, C1, . . ., Cm), then let Cy be the classification that is sought, and Cn all the other categories (C - Cy). Then, one starts with two ranges:

[a,b]—no feature corresponding to elements in Cn has a value in this range.

[a',b']—all features corresponding to Cy have values in this range.

$a' \leq a$ & $b \leq b'$

Call the interval of acceptance, I, where

$I(0) = [a,b]$

$I(1) = [a',b']$

$I(e) = [f(e,a,a'), f(e,b,b')]$

and e is the expectation that the feature exists ($0 \leq e \leq 1$). f is a function that returns a number that is used as a threshold. For example, it can be a step function,

```
f(e,x,y) = if 0.00 ≤ e < 0.33 -> x
              0.33 ≤ e < 0.67 -> (x + y)/2
              0.67 ≤ e ≤ 1.00 -> y
```

linear,

$f(e,x,y) = x + e * (y - x))$,

exponential, or something more complex.

## 4.5.3. Mixed Control Strategies

Since both top-down and bottom-up control can take place, what determines which will be applied and when? A similar question pertains to where the cycle of perception is to be entered. Initially this largely depends on the amount of user direction.

The user can specify any schema(ta) as the starting point for interpretation. If he/she specifies a schema that can be instantiated directly from the data plus some relevant information like a location, then bottom-up processing can start from that point. Otherwise, the schema-specific top-down routine will be initiated. This procedure will send out an appropriate message to some schema lower down in the hierarchy which will propagate until a schema can be instantiated bottom-up from the data. This communication does not necessarily create instances of the higher-level schemata. That is, a schema-specific routine might be evaluated, but if it passes control to another routine without filling in its slots, the data structure itself might not remain.

If the user does not provide any starting place in the network a default node is used. Because it represents general knowledge, one high up in the hierarchies is the most applicable. It would send out model-based top-down messages as in the case above in which the user selects the starting point.

Once a schema is instantiated from the data by having its slots filled then two types of messages can be sent. First, the schema will generally transmit a bottom-up request to higher-level schemata to discover where it should fit into the semantic network. For example, a newly instantiated road will send a message to "road system" which looks for a compatible existing road-system instance to join. If one does not exist, it will be created. Second, the schema may send out suggestions of other possible interpretations. While the road schema was being instantiated, a dark, rectangular region would be a candidate for a car, so the location of the region can be sent to the car schema. These suggestions could go in any direction in the hierarchy--up, down, or laterally.

Once processing has passed the first few stages, there will generally be several possibilities for further processing. These are placed in a priority queue which can be modified by the user (Section 4.4.1). These possibilities include a mixture of both top-down and bottom-up messages[4]. Control will move up and down the hierarchies and at each node more messages will be spawned to induce more processing.

This type of processing is neither top-down nor bottom-up. It takes advantage of the relations between schemata which organizes them into a graph (see also Collins and Loftus, 1975; Bolles, 1979). A message sent from a schema may be going up the hierarchy (bottom-up), down it (top-down), laterally across, laterally across and down, or laterally across and up. There is no name for the latter three modes; the closest might be heterarchical (Winston, 1972). The strategy is best-first or island-driven with the rankings on the priority queue used to determine the order of search. Since hypotheses can be

[4]As will be seen in Chapter 5, there are two types of bottom-up entries, one dealing with the intensity image alone and another that uses the results of sketch map interpretation in conjunction with the image.

retracted, it is more akin to non-monotonic reasoning than most vision systems which usually follow the principle of least commitment (Marr, 1976). It is also described well and is independently motivated by the cycle of perception.

## 4.5.4. Distributed AI, VLSI, and Parallel Processing

Distributed AI is a quite new subset of AI that deals with a group of "distinct, cooperating problem solvers" (Davis, 1980b, p. 42; this report summarizes the various attitudes toward the field). The system being described fits into this model. Each schema can be considered a problem solver that knows how to recognize the object being modelled.

Coupled with recent work in VLSI, exciting possibilities in parallel processing can be imagined. If each schema, or better yet each instance of a schema, had access to its own processor, then much of the interpretation process can proceed in parallel. Instead of messages being queued, they would be sent to active, executing routines. If the number of processors was limited, the priority rankings can be used to determine which schemata would receive attention.

Parallel processing as described would not increase the theoretical power of a vision system only its efficiency. That is, it would not be able to interpret images that were more problematic. However, such a system would make it more convenient for schemata to communicate while they are active. This increased cooperation might create a situation where the interpretation potential is increased.

## 4.6. Using Globals to Remove Thresholds

Thresholds were described in Section 4.5.2 as a necessary evil that will often be found in schema-specific instantiation routines. This section describes a method for reducing the number of magic (ad hoc) numbers that must be used as thresholds.

Take a chain (line) from a sketch map that has been interpreted by sketch map analysis and consider it as a river. If the sketch map is known to be approximately registered to an intensity image of the same scene, then one can use the location of the sketch map chain to search for a river in the image. However, a chain has no thickness and a river does; the chain only approximates the river's location. So, one would probably want to create an ellipse or rectangle around the chain and search within the corresponding region in the intensity image.

The length of the rectangle (or major axis of the ellipse) can be defined as the end points of the chain. However, the width (or minor axis) would have to be an ad hoc number or threshold chosen to reflect what is known about the widths of rivers that are likely to appear in the images to be interpretated. Similar thresholds are required in almost all matches between features from different information sources because the correspondence will always be approximate.

All of the thresholds based on size, such as river width, are related to the scale or resolution of the image in question. If one knows the actual size of each pixel, or a range of sizes, then real-world knowledge of rivers can be used to generate the threshold. For example, if the resolution of the image is 20 feet per pixel and rivers are generally between 5 and 200 feet wide, then the rectangle should be between .25 and 10 pixels wide.

In the situation above, one would choose the larger value (10) to be sure to include enough of the image where the river might be. However, in many cases, one would use the principle of least commitment and choose the smaller value. Higher expectations allow one to relax the thresholds to the larger value as was described in Section 4.5.2. In the terminology described in that section, for this example the ranges are [a,b] = [0,.25] and [a',b'] = [0,10].

Threshold ranges that can be relaxed are applicable even if _ad hoc_ values known only to a schema-specific routine, such as river, are used. The advantage of tying them to the scale of the image is that scale is a global value that holds for all of the schemata in the image. Hence, its value can be factored out of the schema-specific routines. For the above example, the reference might be to "200/resolution", where resolution is a global variable. If all such size-related thresholds are translated into a factor of the scale, then a large number of thresholds are all reduced to one value plus the factors, which are based on model knowledge.

This is not a case of "putting all of one's eggs in a single basket", however, if the principle of graceful degradation is applied. The schema-specific routines can take into account the certainty of the value for resolution whenever it is used, as in generating threshold factors. Then, the quality of that value would determine how good the results were. Such a value's use can be facilitated if scale is represented as a schema, since schemata have a slot for CONFIDENCE plus a mechanism for calculating it (Section 4.2). In addition, schemata have procedures associated with them to respond when the value is added, modified, or removed. There is also a default value that can be used if no value is available. Also, there are attached procedures that can attempt to calculate the value when it does not yet exist. Thus instead of "200/resolution" one might refer to "200/(sgetv 'scale 'feetperpixel 'yes 'one 'default)" in the MAIDS system.

There are two ways to obtain the value of a global such as scale. It can be entered by the user as global input (cf. Section 4.4.1) since scale is often provided with remotely sensed imagery. Or, it can be inferred from the image plus model knowledge. Before, the threshold range was used to help instantiate an object. However, once the object is instantiated, perhaps by some other test or other IS, its known size can be used to limit the possible range of scale values--by multiplying the size by the reciprocal of the factors. This value can be modified and improved, i.e. restricted, as interpretation proceeds. A mechanism for carrying it out is found in the next section.

These global values are somewhat unusual schemata since they are not part of the semantic network of objects. They are similar to global variables in programming languages which differ from the local variables that are lexically ordered by procedure declarations in statically-scoped languages.

Scale is a useful global in reducing size-related thresholds. Many angular thresholds can be related to the azimuth of the sun, especially when shadows are involved. This value is often available with satellite imagery. When dealing with three-dimensional models, the elevation of the sun is also useful.

## 4.7. Cluster Analysis as an Inference Mechanism

In Section 4.5.2, the similarity of pixel intensity values was used to make an inference about the classification of those entities into water and land: namely, similar objects in the scene have similar feature values. Further, if the feature space is selected appropriately, these feature values can be used to discriminate different objects because they will have different clusters of values. If similar, individual features in the image are grouped (clustered), then the corresponding image points are all part of some meaningful class. Derived features of compound objects, such as the orientation of edge segments, are even more useful than pixel intensity values.

Histograms in the proper feature space make clustering apparent. Figure 4.5 shows the edge segments found in an image of Cranbrook, British Columbia, which contains a number of roads. Figure 4.6 is the one-dimensional histogram showing the number of segments having a given orientation. There are two significant sections in this histogram: A corresponds to northeast-southwest roads, and B to northwest-southeast roads. Note however, that the histogram is very rough and that divisions between these sections are often difficult to determine. A smoother diagram would be preferable.

The method whereby image components are translated into a feature space that can be used in discriminating significant objects is known as the Hough transformation. It was originally used to group edge points into line segments by successively clustering one-dimensional histograms of either slopes and intercepts ($y = mx + b$) or angles and distances ($\rho = x\cos\theta + y\sin\theta$) (Duda and Hart, 1972; O'Gorman and Clowes, 1973; Dudani and Luk, 1977). More recently, Ballard and his group have applied the technique to the analysis of intrinsic image data (Ballard, 1981; Ballard and Sabbah, 1981). They used multi-dimensional histograms in a variety of applications to determine model-to-image transformation parameters, surface orientation, optical flow, etc.

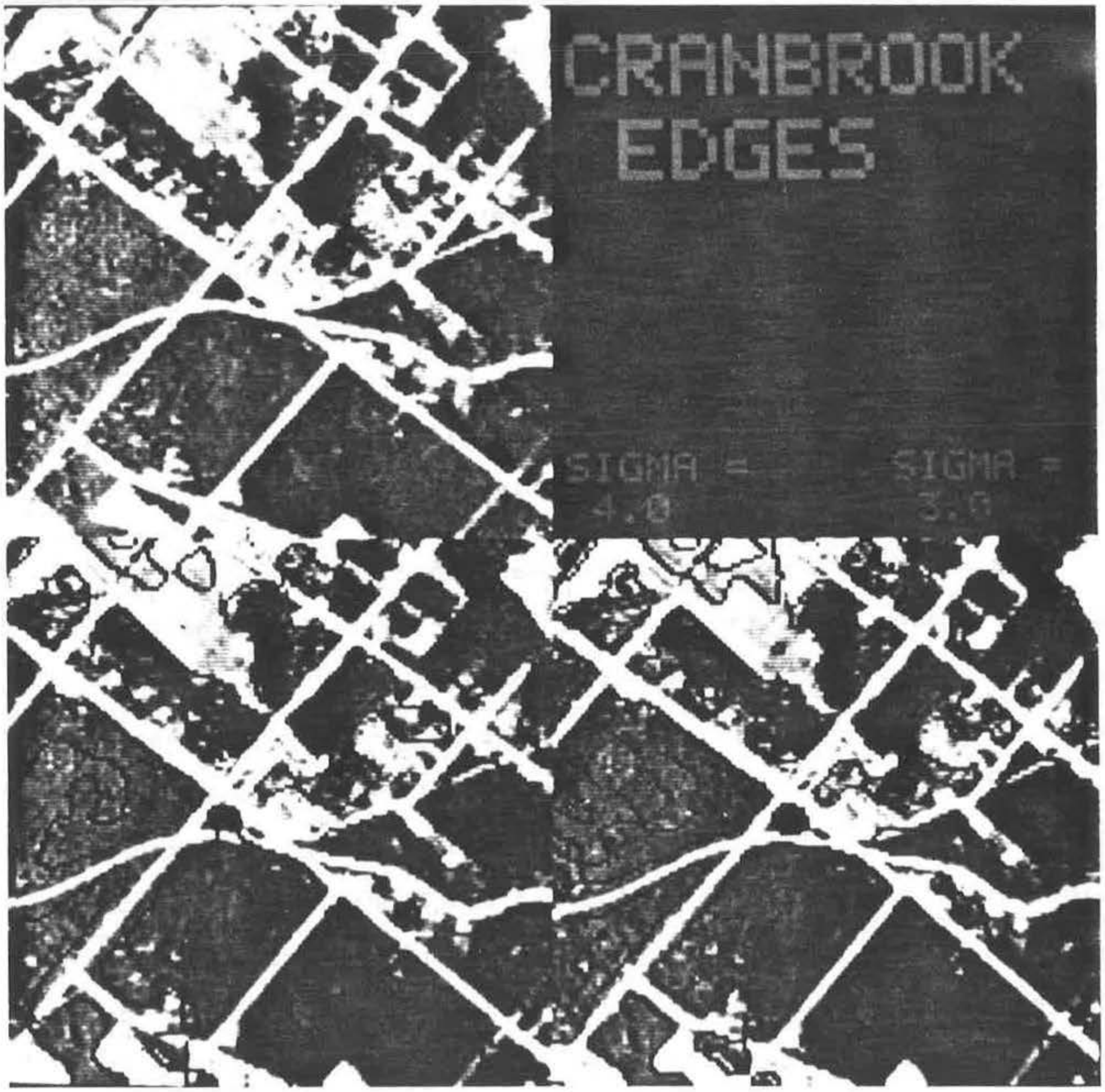


Figure 4.5 Edge Segments Superimposed on Cranbrook

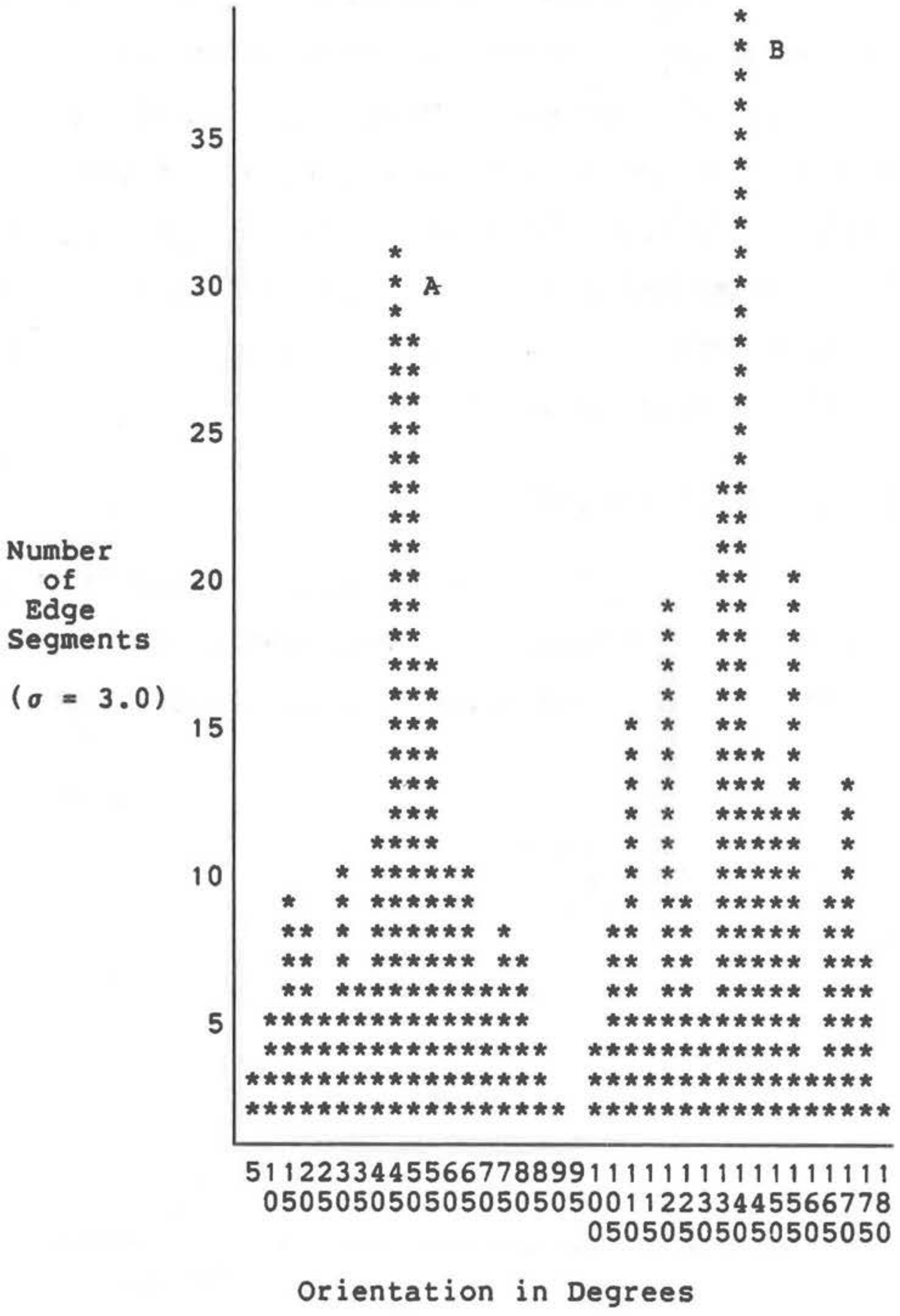


Figure 4.6 A Histogram of Edge Segment Orientations

Clustering is a method of unsupervised, nonparametric pattern recognition. When using histograms, the general strategy uses peaks for the modes of categories and troughs as decision boundaries to classify values. Using image intensities as features, segmentation can be done in this manner (Eigen et al., 1974; Ohlander, 1975; Schacter et al., 1976; Ohta et al., 1978; Tomita, 1981). The method described below follows from this work. Its major advance is the use of Gaussian smoothing to facilitate peak and trough selection.

## 4.7.1. A Method for Clustering

Consider the function in Figure 4.7 as an idealized, continuous histogram. Intuitively, this represents two clusters with modes at peaks A and C and a division at the trough, B. If

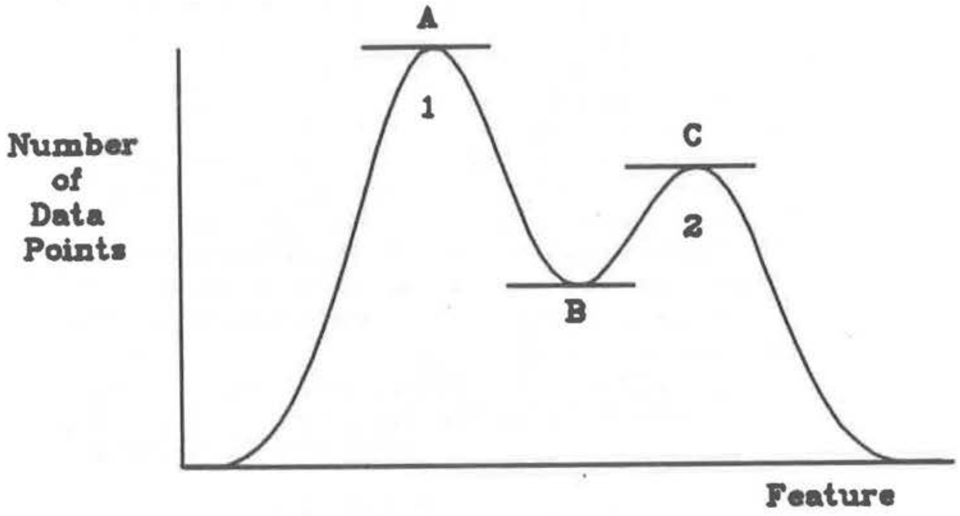


Figure 4.7 An Idealized, Continuous Histogram

we regard the histogram as a function, then notice that points A, B, and C are the only ones (ignoring the ends) where the slope of the tangent to the function (or, equivalently, the first derivative) is 0. Hence, we can find both the modes and decision boundaries by looking for zeros, or zero crossings in the discrete case, in the first derivative of the histogram of a given feature.

This discussion presumes that a histogram can be derived from the data. First, it is clear that n-dimensional histograms can be formed for multivariate samples. Each feature would represent a different dimension. Although, for clarity, the analysis will be explained using the one-dimensional case, it holds in higher dimensions. An example of the two-dimensional case will be given in Section 4.7.5.

The range of feature values can be quite large. However, as long as all the values are finite, or if all the infinite values can be put into a finite number of distinct buckets, then a histogram can be derived. If there are many empty buckets then a suitable transformation (e.g. logarithmic) can reduce the space to a more manageable size or indirect addressing can be used for a computer implementation.

Histograms, by their nature, are discrete, with buckets or bins containing all the values in a certain range. Thus if the data are real-valued, bucket boundaries must be selected: if they are too wide, then too many data points will be put into each bucket and the histogram will be too smooth; if they are too narrow, the histogram will be too rough, or can even flatten out, destroying the modes. These conditions correspond to large bias and large variance, respectively. The first case is a problem that is easily avoided by scanning the data and making sure that no bucket receives more than a small percentage of the total number of samples. The latter case can be avoided if there is some smoothing or grouping of the buckets.

### 4.7.2. Gaussian Smoothing

Many histograms of real-world data have a very noisy appearance, leading to the generation of many tiny local peaks. For this reason, many researchers have employed smoothing techniques before peak selection. In most cases, this has taken the form of local averaging. It has been accomplished either by varying the bin width (Eigen et al., 1974; Leboucher and Lowitz, 1978) or by arithmetic averaging under a moving window (Scott, 1979; Dudani and Luk, 1977).

Uniform averaging is, in effect, a low pass filter. However, following Marr and Hildreth (Marr and Hildreth, 1980; Hildreth, 1980), this filter is not band-limited in the frequency domain. They show that a Gaussian, i.e. normal, function leads to the smoothing filter that is optimal with respect to the tradeoff between spatial localization and maintaining a small variance in the frequency domain.

The one-dimensional Gaussian function,

$$G(x) = 1/\sqrt{2\pi\sigma^2}\,\exp(-x^2/2\sigma^2)$$

is depicted in Figure 4.8a. Since the scale facter $1/\sqrt{2\pi\sigma^2}$ is irrelevant to the following, it will be omitted.

There is one required parameter, $\sigma$, which must be supplied before the function can be calculated. A heuristic will be provided in Section 4.7.4 for divining good values for $\sigma$.

## 4.7.3. Peak and Trough Detection

Once the histogram is smoothed, clusters can be separated by taking derivatives of the histogram function. Specifically, the zero crossings of the first derivative will indicate the peaks and troughs and the second derivative can be used to distinguish one from the other.

Figure 4.9a shows an example histogram and Figure 4.9b shows the result of convolving it with a Gaussian filter with $\sigma = 5$. The first and second derivatives are shown in Figure 4.9c and d.

The zero crossings in Figure 4.9c show the points of interest. C2 and C4 are the modes of the clusters and C1, C3, and C5 are the decision boundaries or cutoff points for membership in those clusters. The second derivatives in Figure 4.9d show that peaks have a corresponding negative value in the second derivative and troughs have a positive value.

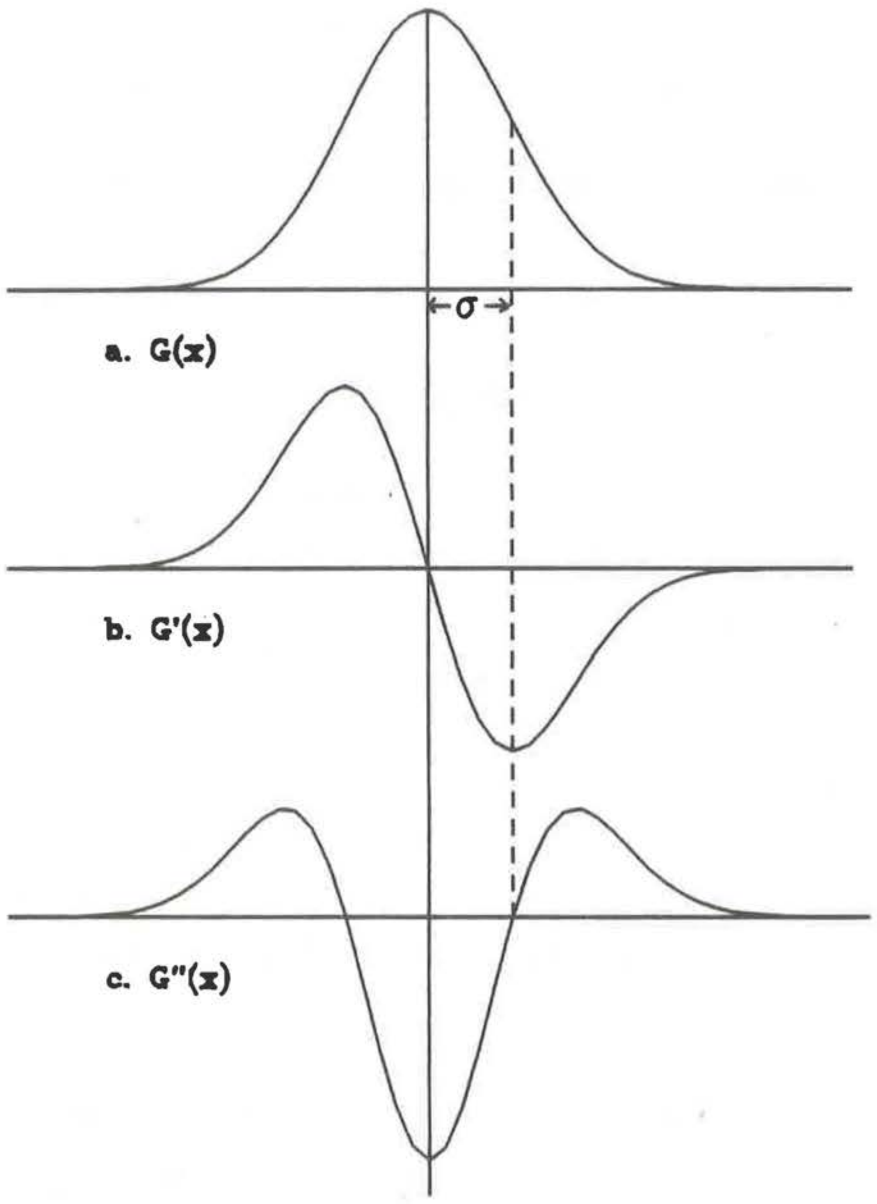


Figure 4.8 The Gaussian Distribution
a. G(x) The Gaussian function
b. G'(x) The first derivative of the Gaussian function
c. G''(x) The second derivative of the Gaussian function

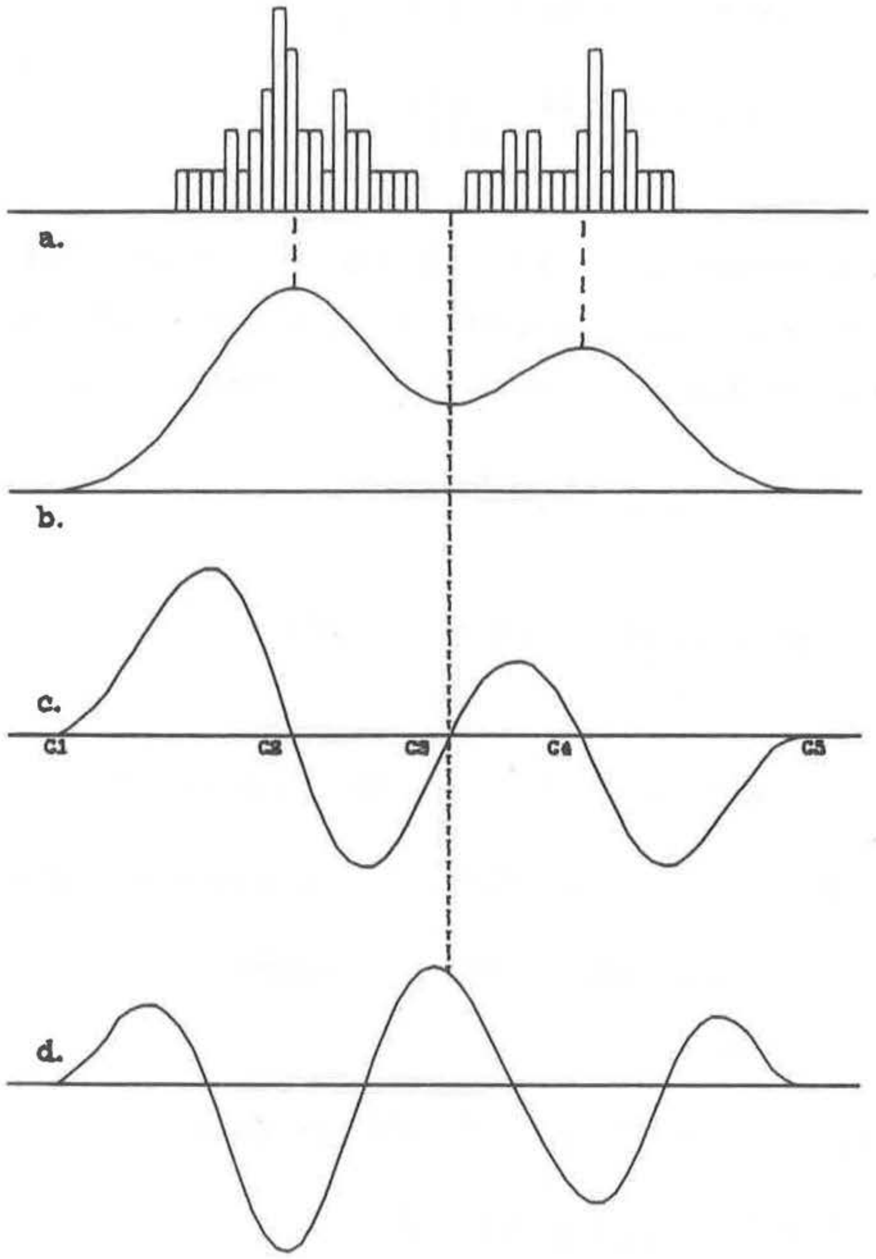


Figure 4.9 A Gaussian-Smoothed Histogram and Derivatives
a. Histogram
b. Histogram convolved with the Gaussian function
c. Histogram convolved with the first derivative of the Gaussian function
d. Histogram convolved with the second derivative of the Gaussian function

Thus, finding the zero crossings in

$$f(x) = \frac{d[G(x) * H(x)]}{dx}$$

is the equivalent of peak and trough selection where G is the Gaussian, H is the histogram function, and * is the convolution operator. By the derivative rule for convolutions,

$$f(x) = \frac{dG(x) * H(x)}{dx}$$

where, $\frac{dG(x)}{dx} = -x \exp(-x^2/2\sigma^2)$

and exp stands for e raised to a power.

Similarly, the second derivative of the Gaussian function is

$$\frac{d^2G(x)}{dx} = (x^2-\sigma^2)\exp(-x^2/2\sigma^2).$$

These are plotted in Figures 4.8b and 4.8c.

## 4.7.4. The Stability Heuristic

The method described for edge detection is dependent on the amount of smoothing. This is determined by the value chosen for $\sigma$. A large $\sigma$ will cause many points to be merged together, hence fewer peaks and fewer clusters. Conversely for small $\sigma$ there are potentially as many clusters as data points.

These conditions are shown in Figure 4.10, for the same data used for Figure 4.9. The value $\sigma = 10$ in Figure 4.10a leads to only a single peak, whereas $\sigma = 1$ in Figure 4.10b results in 4 peaks.

Eigen et al. (1974) and Postaire and Vasseur (1981) both describe clustering methods that depend on a quantization factor (their $\delta$ and M respectively). As is the case with $\sigma$, the value for the quantization factor determines the number of resulting clusters. By plotting the number of components or clusters

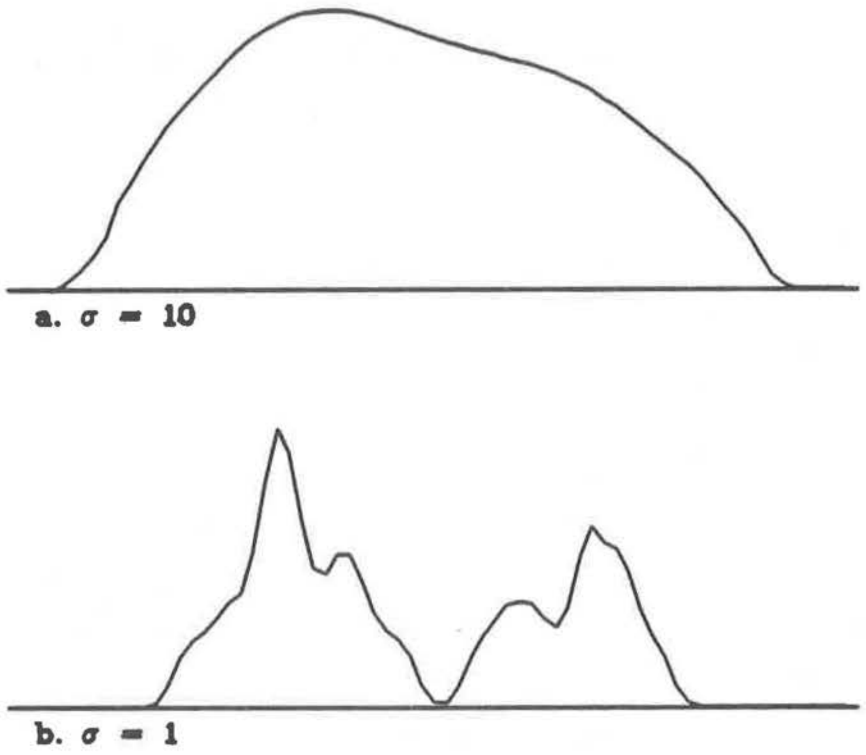


Figure 4.10 The Effects of $\sigma$ on Smoothing
a. $\sigma = 10$
b. $\sigma = 1$

versus $\delta$ (or M), they discovered a region of stability--the largest range of $\delta$ (or M) values where the number of clusters remained constant. Postaire and Vasseur found that this value of M was very close to the optimal Bayes minimum error rate.

This heuristic works very well for the parameter $\sigma$. Figure 4.11 shows the result of various $\sigma$'s on the number of clusters for the example histogram of Figure 4.9. The longest stable region extends from $\sigma$ equal to 2 to 8. The midpoint of this region ($\sigma = 5$) is the value used in the Gaussian smoothing of Figure 4.8 and 4.9.

There is one immediate difficulty involved with using this heuristic that concerns large values of $\sigma$. As one increases $\sigma$,

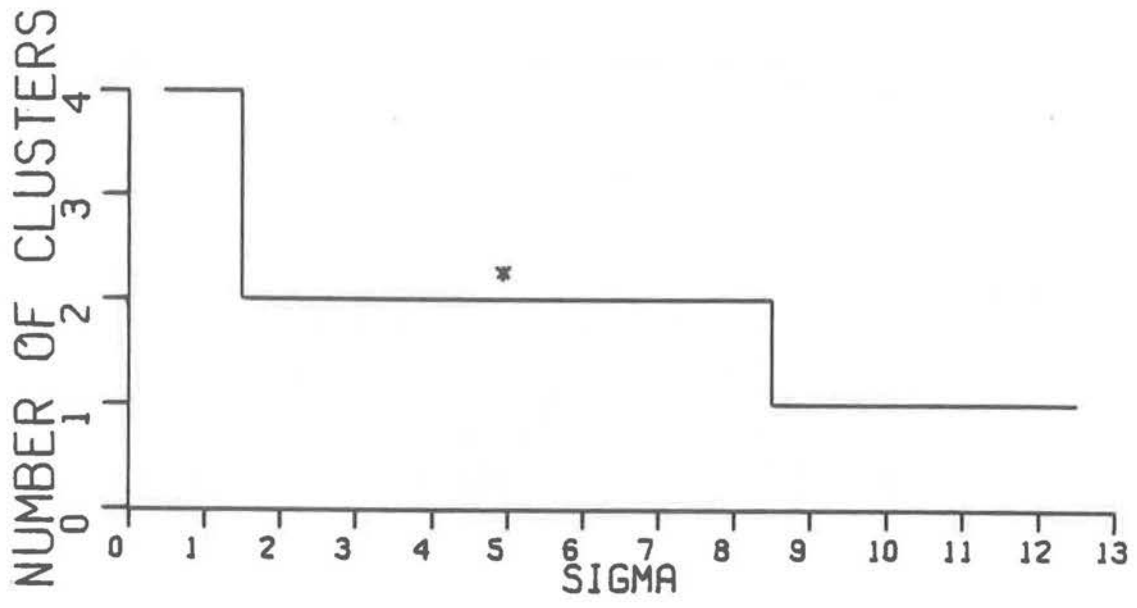


Figure 4.11 Finding the Largest Stable Region of $\sigma$ Values

eventually there will be only one cluster left. All larger $\sigma$'s will also result in one cluster. Consequently, there will always be an infinite range of $\sigma$'s leading to one cluster. If there is no a priori way to fix an upper limit to the value of $\sigma$, one must ignore one cluster as a valid result of the heuristic.

## 4.7.5. Extensions to Multivariate Data

As one goes from univariate to multivariate data spaces the basic technique remains the same: smoothing and differentiation are accomplished by convolution, then the modes and decision boundaries are found, although this becomes more complex. Finally, the stability heuristic is applied. A bivariate example will show how the generalization takes place.

Figure 4.12 shows a two-dimensional example with five clusters created by a pseudo-random number generator. Table 4.2 shows the means, covariances, and number of samples in each cluster.

The first step in the procedure is to convolve this histogram with a suitable filter. The two-dimensional Gaussian function (without the scale factor) is

$$G(x,y) = \exp(-(x^2 + y^2)/2\sigma^2).$$

To avoid directional effects, one would like to use non-directional derivatives. There is no non-directional first derivative; however, for the second derivative, we can use the

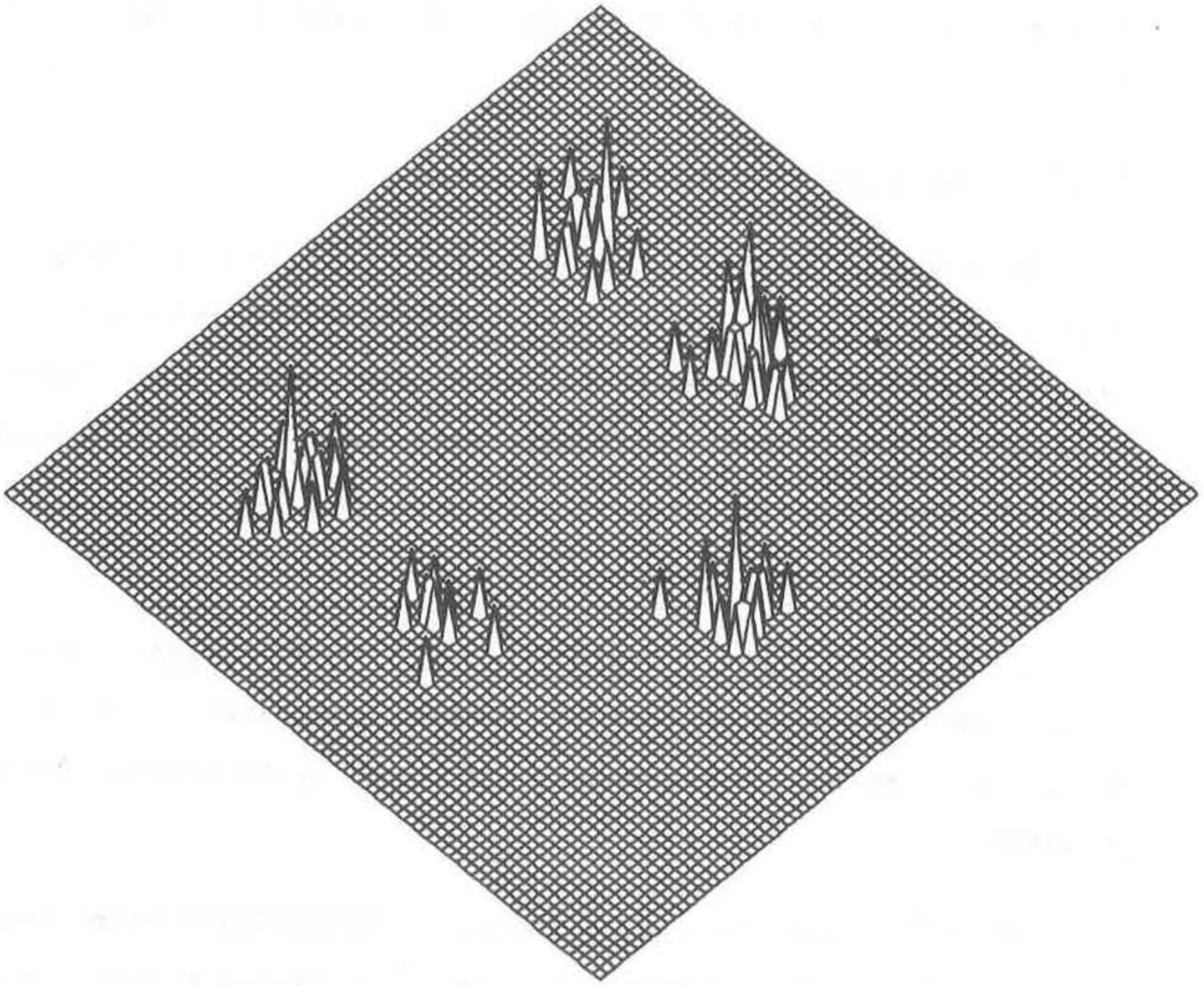


Figure 4.12 A 2-D Histogram of a Bivariate Distribution

| CLUSTER | MEAN | COVARIANCE MATRIX | SAMPLE SIZE |
|---|---|---|---|
| 1 | [20 60] | $\begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix}$ | 20 |
| 2 | [60 60] | | 20 |
| 3 | [20 40] | | 10 |
| 4 | [60 40] | | 30 |
| 5 | [40 20] | for all clusters | 20 |

Table 4.2 The Relevant Statistics for Figures 4.12-17

Laplacian:

$$\frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} =$$

$$\nabla^2 G(x,y) = (x^2 + y^2 - 2\sigma^2)\exp(-(x^2 + y^2)/2\sigma^2).$$

The results of convolving the histogram with these two filters with $\sigma = 4.5$ can be seen in Figure 4.13 and 4.14.

Although there are no first derivative zero crossings, the number of clusters can be found from the Gaussian-smoothed image. A small, local operator can be used to find the local maxima in the image. Each peak corresponds to a cluster. Figure 4.15 shows the zero crossings of the second derivative function along with the maxima of the smoothed image (the +'s).

As in the one-dimensional case, the number of modes depends on the amount of smoothing. The number of peaks can be used with the stability heuristic to find a good choice for $\sigma$. This is done in Figure 4.16, which shows how a value of $\sigma = 4.5$ gives 5 clusters.

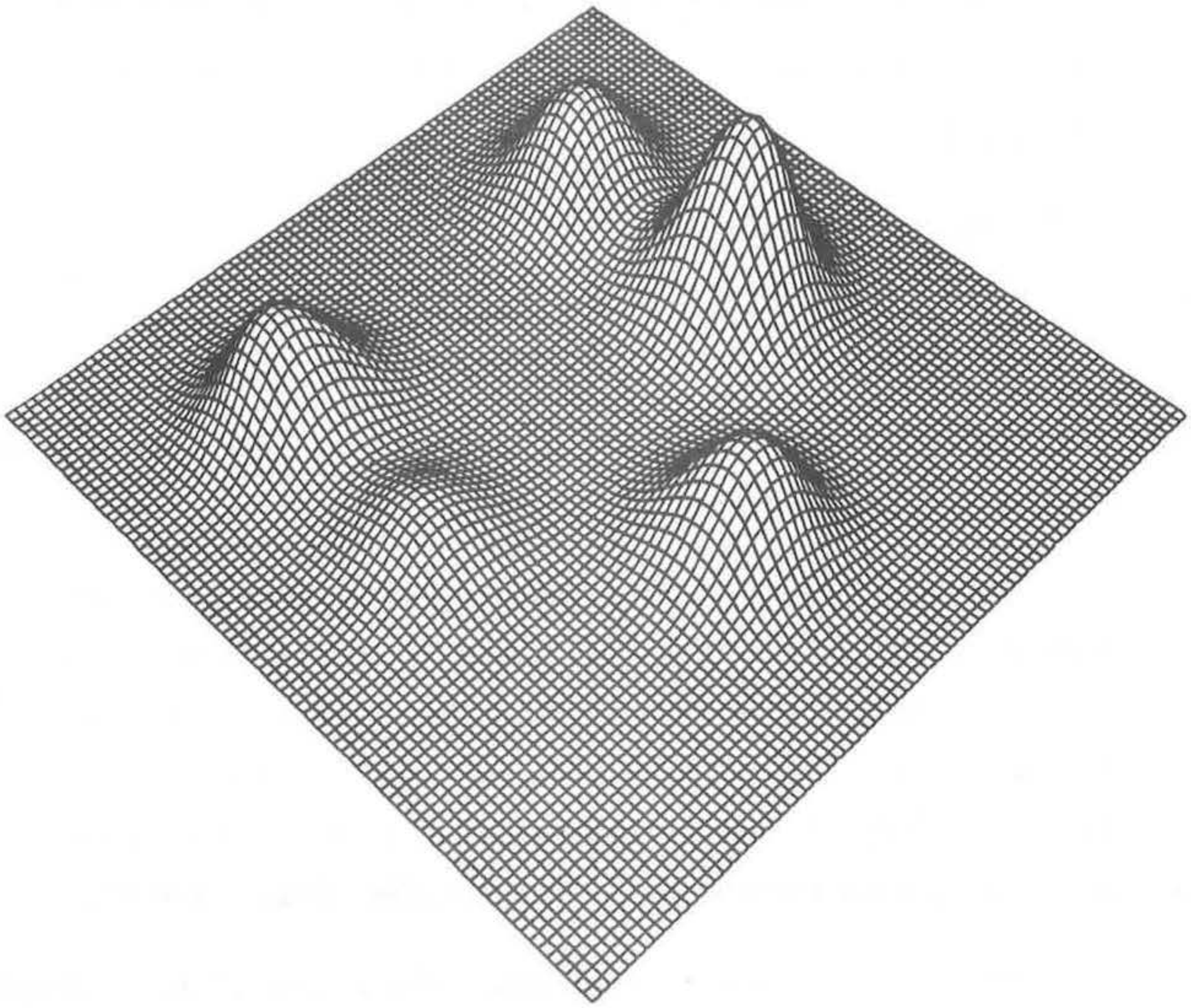


Figure 4.13 The Histogram Convolved with the Gaussian function, where $\sigma = 4.5$.

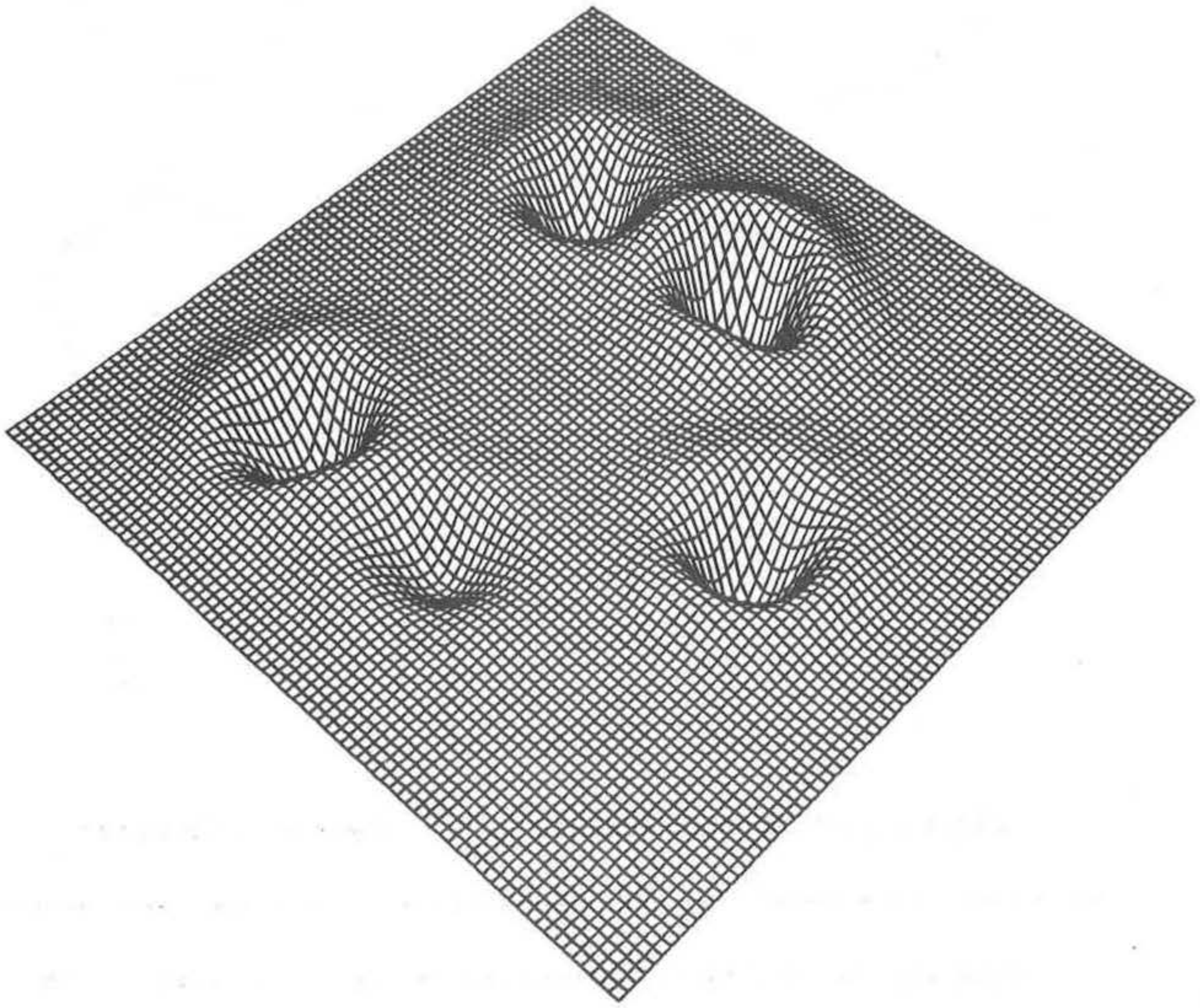


Figure 4.14 The Histogram Convolved with the Laplacian of the Gaussian function

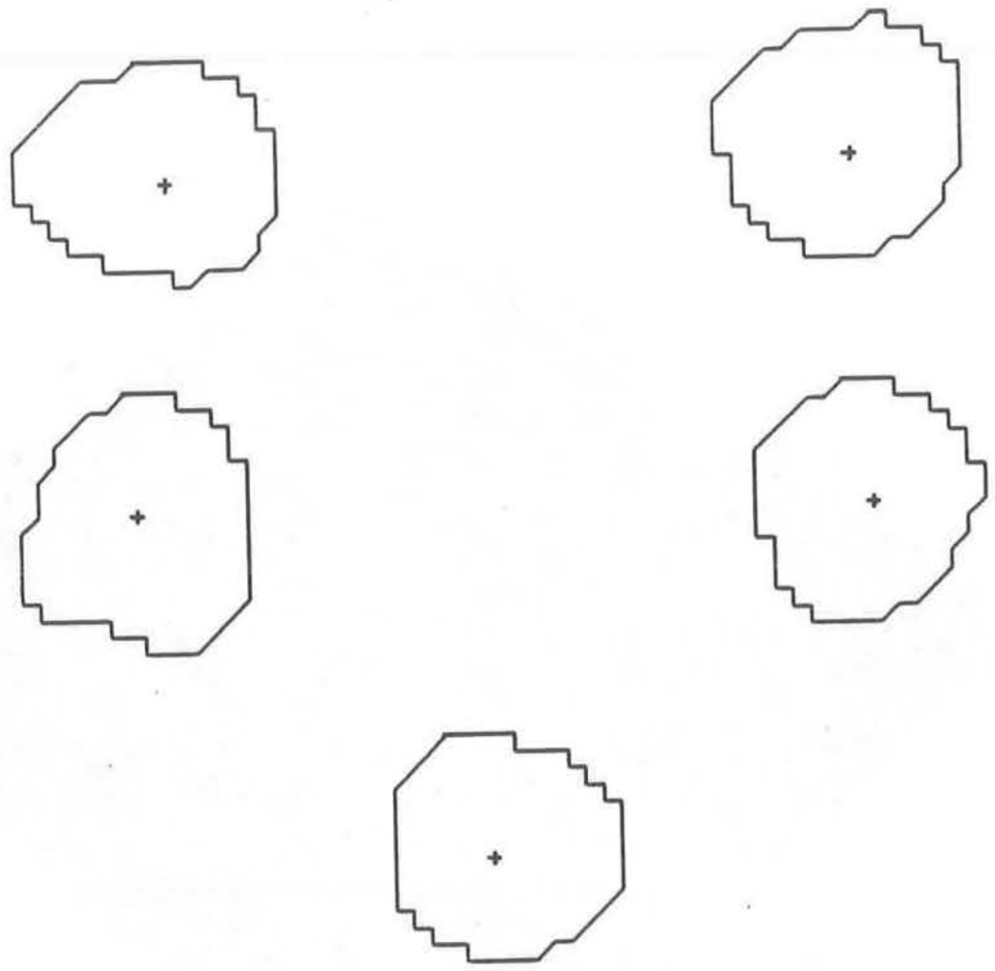


Figure 4.15 Zero Crossings of the Second Derivative

Finding the decision boundaries is not as simple in two dimensions. The second derivative zero crossings correspond to inflection points on the original smoothed convexities. If the distributions are normal and rotationally symmetric, 67 percent of all the points in the cluster will fall into the portion where the second derivative is negative. Hence the decision boundaries will be somewhere between these negative second

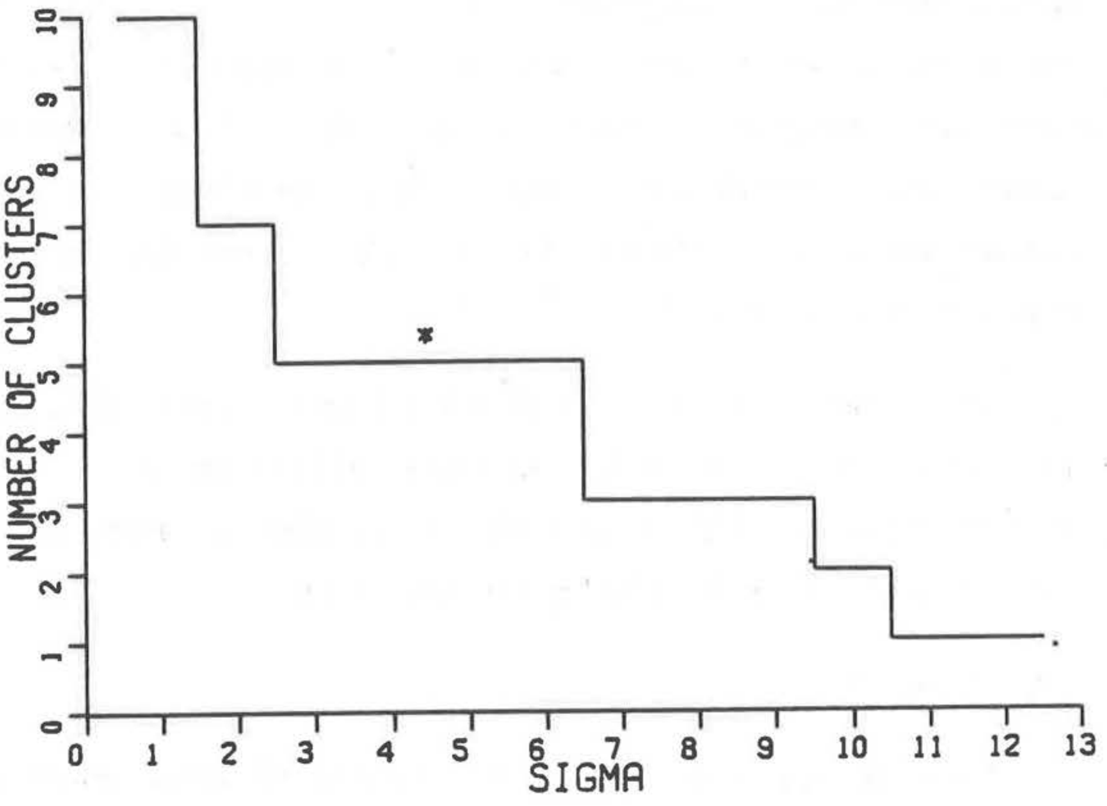


Figure 4.16 Determining $\sigma$ Using the Stability Heuristic

derivative regions (see Figure 4.15). One can grow the zero crossing boundaries for all the clusters until they meet, then use the resulting lines as decision boundaries.

One alternative scheme is to use some form of nearest neighbour technique for points in regions having a positive second derivative. The prototype for each cluster would be its peak. Similarly, a maximum likelihood technique might use the height of the peak as the a priori probability for a cluster.

Another alternative is to treat the smoothed histogram as a three-dimensional topographic surface. If the clusters are thought of as peaks, then the decision boundaries will fall along the channels. Using suitable definitions, channels and ridges, which cross at passes, form complementary networks (Fowler and Little, 1980). The channel network for this example is found in Figure 4.17.

While the nature of the problem becomes more difficult in two dimensions, the basic technique still applies. The extension to three and higher dimensions can also be made, but at the cost of greater complexity and computation.

## 4.8. Summary

This chapter has introduced a number of tools which can be put together into a vision system. Such a system, called MISSEE, is the subject of the next chapter. The techniques described in the introduction are designed to achieve the goals of descriptive and procedural adequacy and graceful degradation. How well those criteria have been met will be discussed in Chapter 6 which includes the results of running MISSEE on several examples.

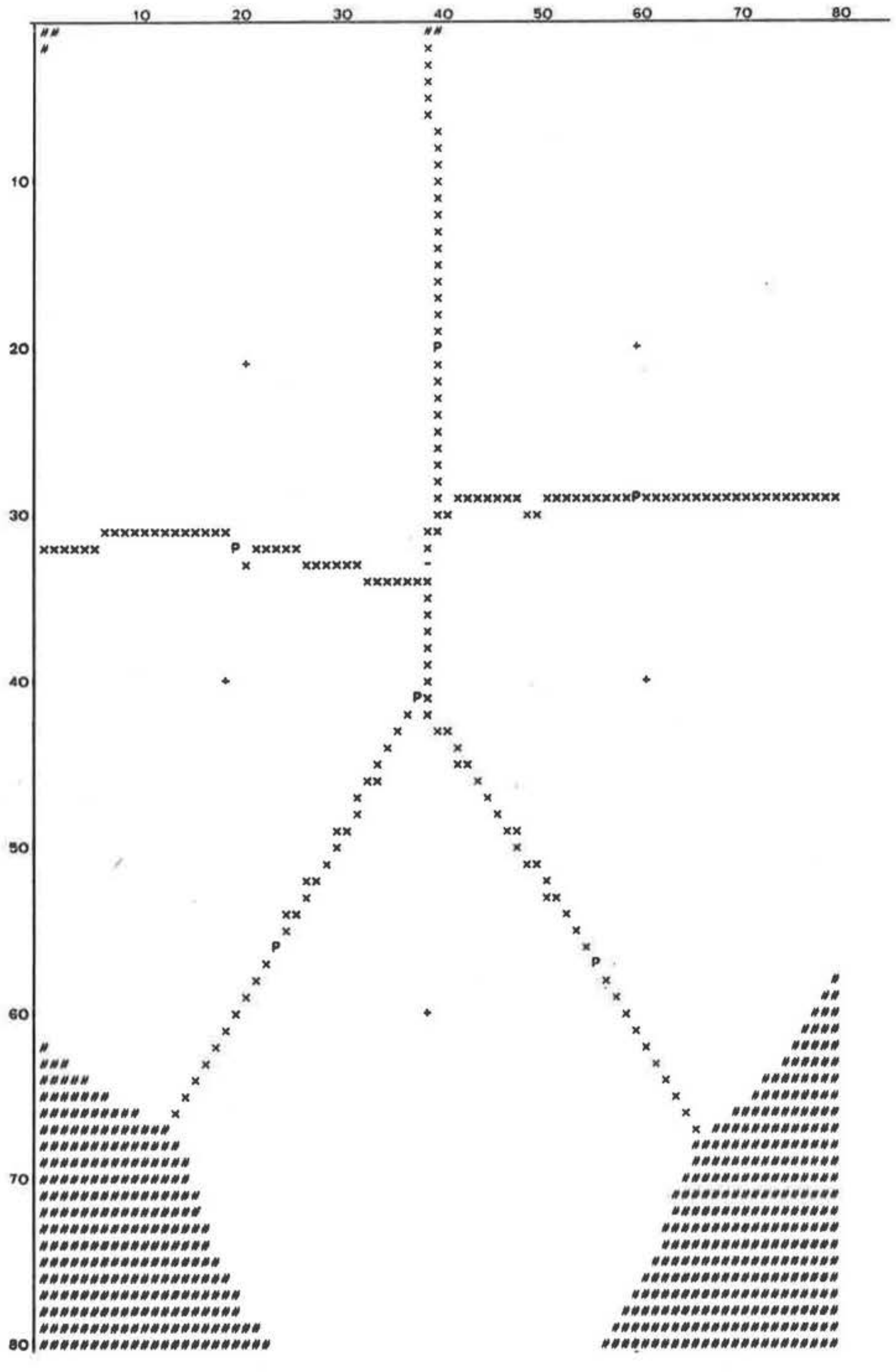


Figure 4.17 Using Channels as Decision Boundaries

| | | |
|---|---|---|
| Legend | + | Peak |
| | - | Pit |
| | P | Pass |
| | x | Channel |
| | # | Zero value of the Surface |

# CHAPTER 5

# The Implementation

## 5.1. Introduction

The MISSEE (Multiple Information Source SEE) system is an implementation of many of the ideas described in the previous chapter. Figure 5.1 shows an overview of the components of MISSEE. The core of the system is a semantic network of schemata which both controls the interpretation process and builds the resulting representation. The schemata are created and utilized by the schemata manipulation subsystem (called MAIDS--MISSEE Aids). The three input information sources that feed into MISSEE are a digitized image, a sketch map, and the user.

The system has gone through two incarnations. Originally it was written in Multilisp (Koomen, 1980) and Pascal and ran on an Amdahl 470. It currently exists in a Unix environment on a VAX 11/780 utilizing Franz Lisp and C.

MISSEE is not designed to be an end product. It was written to test the ideas of the previous chapter in a domain that was restricted, yet rich enough so that all of the problems in model-based interpretation can arise (cf. Section 3.2). While not everything proposed in Chapter 4 has been implemented, a subset demonstrating the power of the key ideas has been. The following sections will describe what has been implemented and

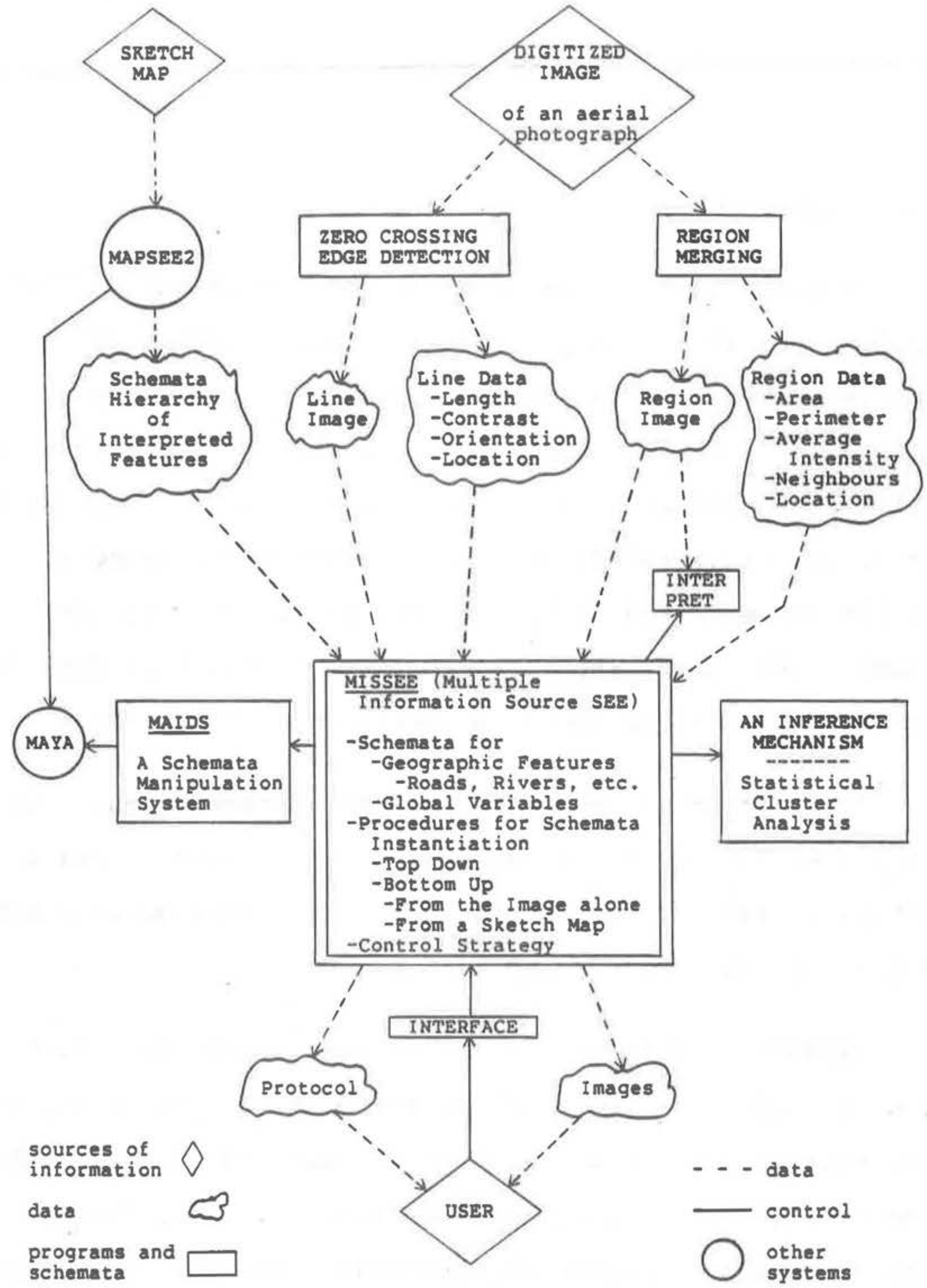

SKETCH MAP
DIGITIZED IMAGE of an aerial photograph
MAPSEE2
ZERO CROSSING EDGE DETECTION
REGION MERGING
Schemata Hierarchy of Interpreted Features
Line Image
Line Data -Length -Contrast -Orientation -Location
Region Image
Region Data -Area -Perimeter -Average Intensity -Neighbours -Location
INTER PRET
MAYA
MAIDS A Schemata Manipulation System
MISSEE (Multiple Information Source SEE)
-Schemata for -Geographic Features -Roads, Rivers, etc. -Global Variables
-Procedures for Schemata Instantiation -Top Down -Bottom Up -From the Image alone -From a Sketch Map
-Control Strategy
AN INFERENCE MECHANISM Statistical Cluster Analysis
INTERFACE
Protocol
Images
USER
sources of information
data
programs and schemata
data
control
other systems

Figure 5.1 The MISSEE System

point out what has not.

## 5.2. MAIDS--MISSEE Aids

MAIDS is an implementation of the schemata manipulation system described in Section 4.2. Written in Franz Lisp as an extension of Maya (Havens, 1978), it makes available a number of Lisp functions for the creation, modification, manipulation, and output of schemata. A user manual describing the use of these functions is found in Appendix A.

While MAIDS is a general purpose system for handling schemata, MISSEE contains specific schemata in the geographic domain that are useful in analyzing aerial photographs. These include such objects as roads, rivers, bridges, road systems, mountain ranges, and geosystems. The system contains stereotypic, general information in the form of slots, or attributes, which define the object and possible default values, and attached procedures which fill those slots during instantiation--see section 5.4.

General objects, such as road, are distinguished from particular instantiations of them (road-4) via the INSTANCE link and its inverse, AIO, an instance of. The instances are created during instantiation of a particular image, and, being hypothetical, may be destroyed if insufficient or contrary evidence is found regarding the entity. Objects, being part of general knowledge, are rarely, if ever, destroyed.

Two of the major organizing relationships within both objects and instances are specialization and decomposition. Specialization (SPECIALIZES-TO/AKO) and decomposition (DECOMPOSES-TO/APO) form hierarchies that are a natural way to organize knowledge.

The previous three relations are all part of the standard MAIDS system because of their general usefulness in schemata organization. A fourth relation, "neighbours", (its inverse is also neighbours) has been added specifically for the MISSEE system. It is used to denote spatial congruency in a particular scene. Thus if it is found that river-4 flows under bridge-2, then river-4 will include a NEIGHBOUR LINK to bridge-2 and vice versa. This fact is used to determine which schemata will be part of higher level instances in the specialization and decomposition hierarchies. That is, river-4 and bridge-2 will be part of the same river system instance (for example, river-system-1).

The specialization and decomposition relations form hierarchies that are present in both general and particular knowledge. Hierarchies define an up-down positioning of entities. The instance and neighbour relations do not form hierarchies and only relate instances (particular knowledge). They can be thought of as defining the left-right positioning of schemata.

Thus the initial MISSEE system contains generic objects formed into specialization and decomposition hierarchies. During instantiation, those schemata specific to the image being interpreted are represented in a parallel network of instances and related by AIO LINKS to the general objects. Additionally, this semantic network is organized by specialization and decomposition and uses the neighbour relation to determine which instances should be grouped as parts or kinds of higher level entities.

CONFIDENCE values are used in a very simplistic manner in MISSEE, again not taking full advantage of the MAIDS capabilities. The results of interpretation cause some minor variations in the reliability of instantiated schemata, but the basic precedence is as follows: schemata instantiated with the aid of the sketch map are given more credence than those instantiated from the intensity image alone. And, higher level schemata (that is, higher in either the decomposition or specialization hierarchies) are given higher confidence than schemata lower down. Information provided by the user is generally accompanied by a confidence value. This ordering is done to ensure that the system will exhibit graceful degradation.

Procedures are attached to slots, the CONFIDENCE values, and the schemata as a whole. In MISSEE, schemata have two types of attached procedures: top-down--to interpret messages received from above; and, bottom-up--either to respond to a message from below or to interact directly with an IS. There are two types of bottom-up procedures for handling input data: those concerned with the intensity image alone and those that can use the results of sketch map analysis to guide interpretation. A third possibility that was not implemented in MISSEE (due to a lack of interactive graphics capabilities) is a set of procedures that interact directly with the user at this level. As will be seen in Sections 5.3 and 5.6, the user interface occurs elsewhere.

## 5.3. The Input Information Sources

The goal of MISSEE is to interpret digitized, monochromatic aerial photographs of small urban scenes. Thus the primary IS is an intensity image derived from a photograph, such as Figure 5.2 of Ashcroft, British Columbia. This photograph is a portion of Figure 1.1. Two other input IS's are available to help with the interpretation of the intensity image: a sketch map and direct guidance from the user. This section describes the information specific to the three IS's.

### 5.3.1. The Digitized Image

Most work on segmenting images has concentrated on either edge detection or region merging. While they may be considered dual representations (edges are based on intensity discontinuities whereas regions are based on the homogeneity of intensities), each emphasizes different aspects of the image. Edges pertain to the location and orientation of surface boundaries while regions are more relevant to surface shape and connectivity. In accordance with the hypothesis of this dissertation (the more different, the better), both types of information from

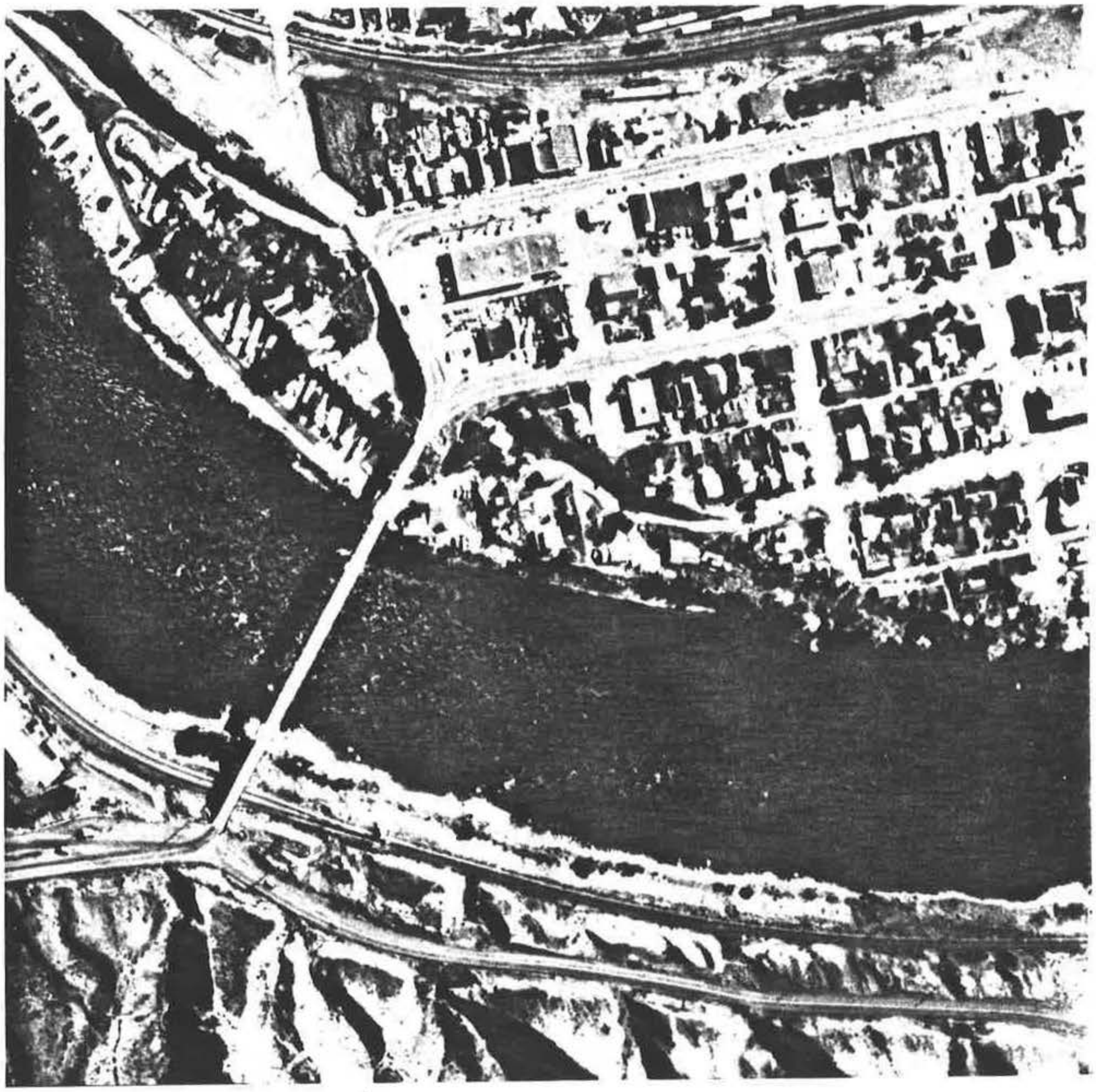


Figure 5.2 Ashcroft, British Columbia

the digitized image are made available[1].

### 5.3.1.1. Edge Detection

A Marr-Hildreth edge detector was implemented that finds zero-crossings of the Laplacian of the Gaussian of the intensity image (Marr and Hildreth, 1980; Hildreth, 1980). The particulars of the implementation are described in Appendix B.

The end result of the edge detection process is a "line image" that corresponds to the intensity image except that instead of intensities, each pixel contains a value specifying whether or not it is an edge point. Figures 5.3b and 5.3c show the edge segments overlaid on the image of Ashcroft for $\sigma$ values of 1.1 and 2.2 respectively. $\sigma$ is related to the amount of detail recovered by the edge detector: the smaller the $\sigma$ value, the greater the resolution.

Further processing is carried out on each edge segment to generate a version of the raw primal sketch (Marr and Hildreth, 1980). For each edge segment, information is stored about its location, length, and orientation. A final value, which we shall call contrast, indicates how large the discontinuity in intensities on either side of the edge is. It is a first derivative taken from the original image.

[1]See (Nevatia and Price, 1978 and 1982) and (Nazif and Levine, 1982) for other systems that combine edge and region information.

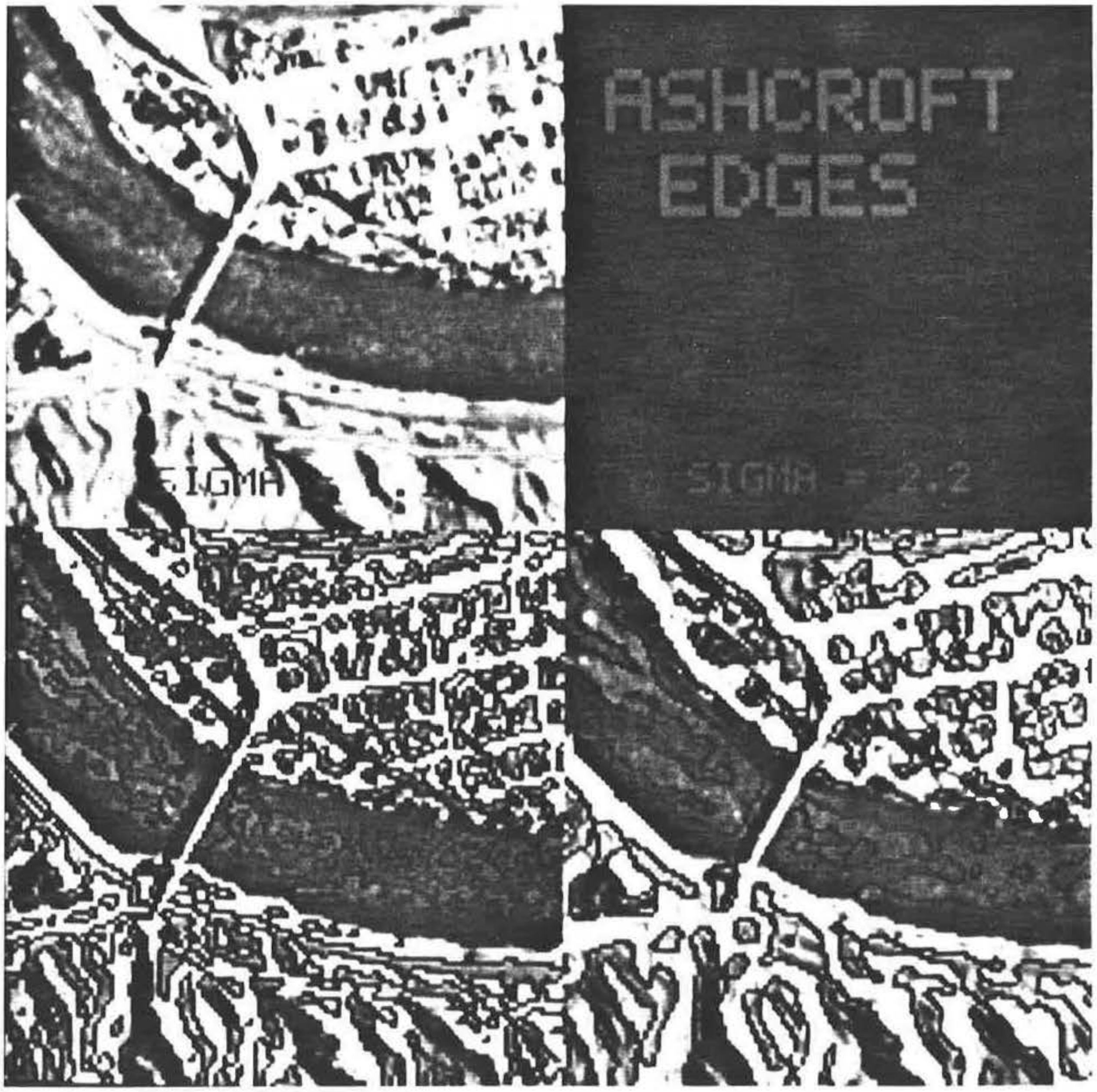


Figure 5.3 Ashcroft: Edge Detection

a. (upper left) Ashcroft
b. (lower left) edges from $\sigma$ = 1.1
c. (lower right) edges from $\sigma$ = 2.2

The zero-crossing edge detector responds to all edges in an image regardless of their significance. Contrast and length values (along with model knowledge) are useful in ordering the edge segments. Figures 5.4 and 5.5 again show edge segments derived from Ashcroft using different $\sigma$ values. Each subfigure shows the edges ranked by a different criterion. The order is indicated by the colour of the segment: bright blue to dark blue is the highest rank, followed by bright to dark red. It is evident that no single criterion ranks all the "significant" edges at the top. Model knowledge can be used, however, to determine the proper criteria for individual objects.

#### 5.3.1.2. Region Merging

Region merging is accomplished by a method described by Freuder called affinity (Freuder, 1976). Basically it merges regions whose average intensities are most similar. Details can be found in Appendix C. Although quite a simple algorithm, it produces results reasonable in appearance. Its most prominent fault is that no concern is paid to the shape or length of common boundaries between regions which leads to some "incorrect" merges (see Brice and Fennema, 1970). Segmentation based on simple, unreliable techniques was used because 1) even the best segmentation techniques render less than perfect results, 2) it shows how cooperative interpretation is powerful enough to overcome the deficiencies of a simple segmentation technique, and 3) it is easy to implement.

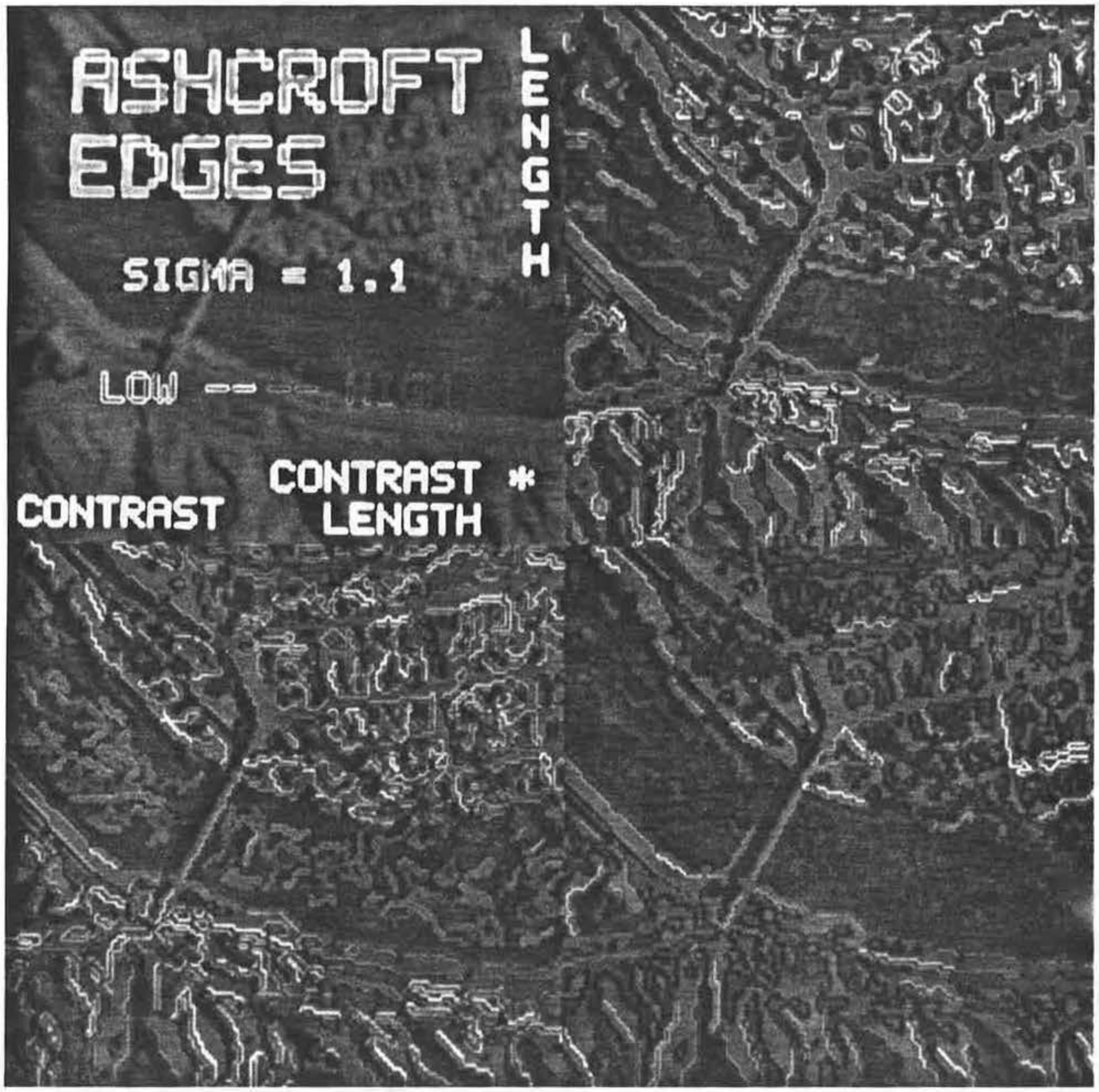


Figure 5.4 Ordering Edges in Ashcroft, $\sigma = 1.1$

a. (upper right) ordered by length
b. (lower left) ordered by contrast
c. (lower right) ordered by contrast * length

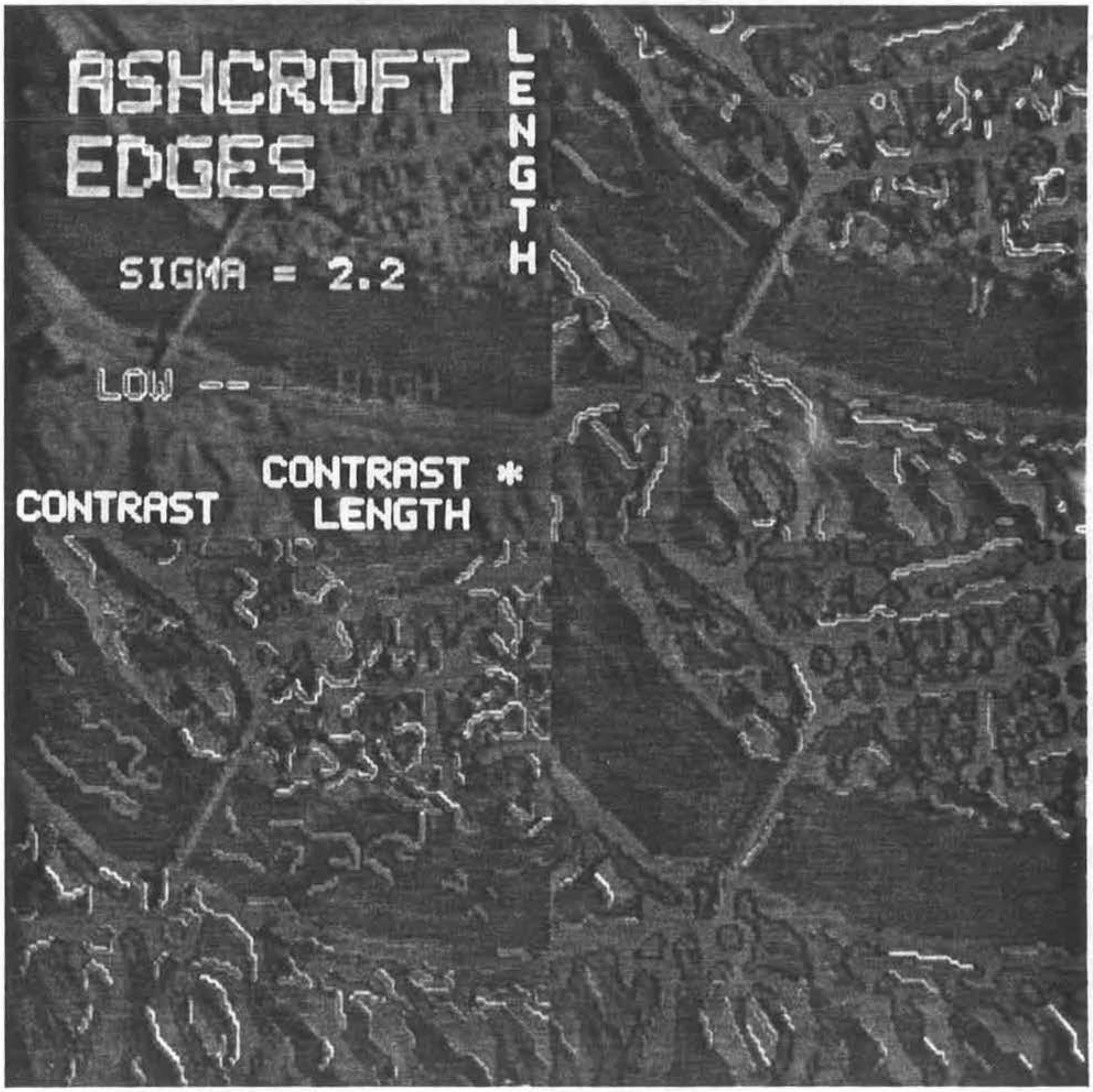


Figure 5.5 Ordering Edges in Ashcroft, $\sigma = 2.2$

a. (upper right) ordered by length
b. (lower left) ordered by contrast
c. (lower right) ordered by contrast * length

Figure 5.6 shows the results of region merging on the Ashcroft image stopped when there are 172 and 75 regions left. The process can either be stopped by the user or by the algorithm when there are no potential merges left. In this case, it was stopped by the user. The result of this process is a "region image" where each pixel indicates to which region it belongs.

As with edges, post-processing is carried out to derive primal sketch-like information for each region. The location of a perimeter point is stored, along with the length of the perimeter and the area of the region. The average intensity of all the pixels in the region is calculated as part of the affinity measure. Finally, the list of all neighbouring regions is made explicit.

Both edge detection and region merging are local operations that can be carried out on subimages of any size. Both are fairly expensive computationally. The former involves convolution and the latter many iterations: one for each stage of merging. If model knowledge can be utilized effectively, it may be possible to limit the parts of the image where edge detection and/or region merging must be performed. For example, if it is known or discovered that a large portion of a particular image contains no features of interest, then it can be ignored. Or, small sections of the image may be searched in order of likelihood for an object of interest and when it is found the search could terminate. The only new (but not necessarily major) problem this strategy entails is how to "put together" the

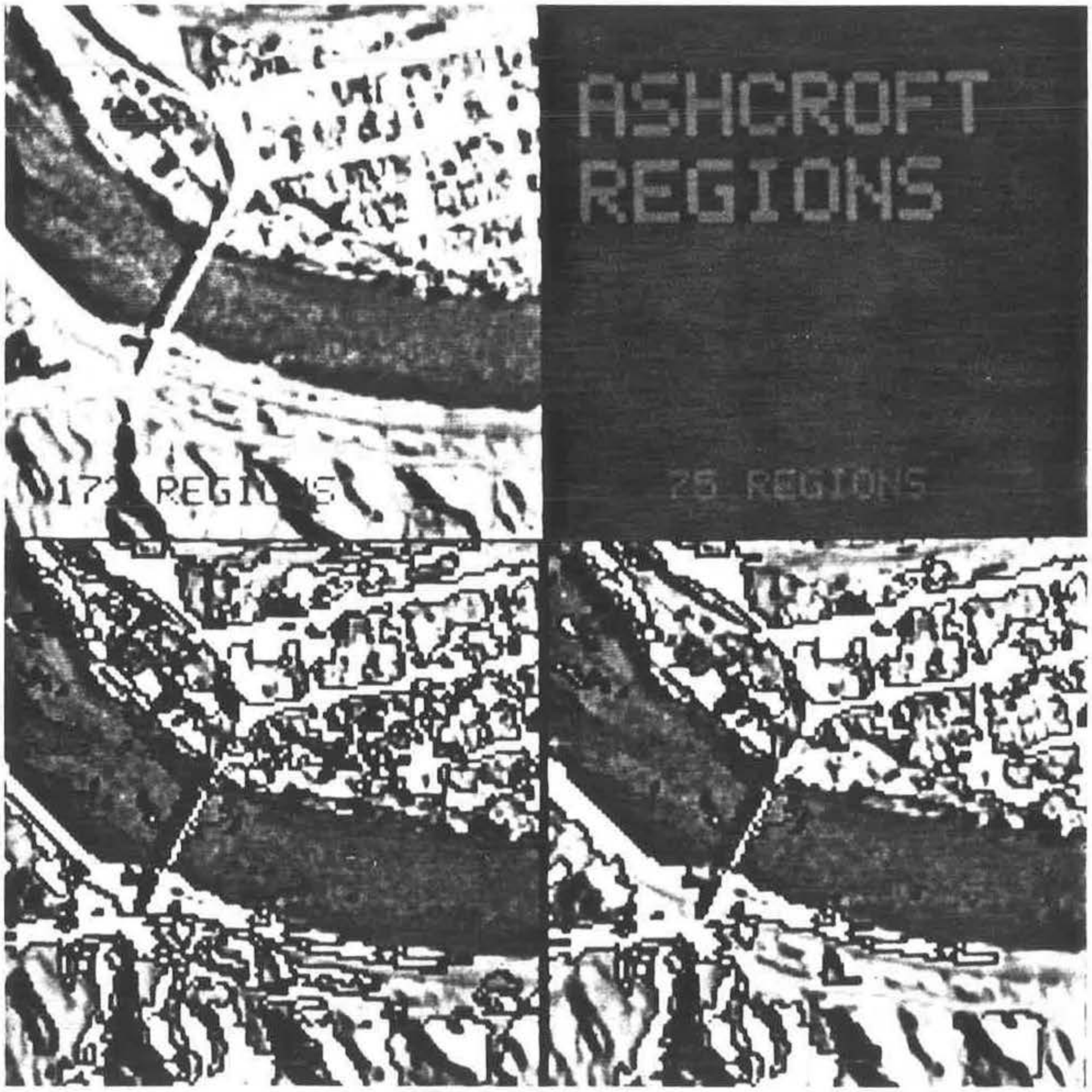


Figure 5.6 Ashcroft: Region Merging

a. (upper left) Ashcroft
b. (lower left) 172 regions
c. (lower right) 75 regions

information of two contiguous subimages at their mutual boundary.

This methodology can increase the efficiency of the overall interpretation greatly if only a small portion of the whole image needs to be examined. On the other hand, it would not increase the problem solving power of the system. It has not been utilized in the current implementation because of the difficulty in coordinating the various programs and data. For example, the region merging process is tied to a specific output device (so the user can decide when to stop the merging) that was not connected to the computer used to run MISSEE.

## 5.3.1.2.1. A Categorization for the Regions

A crude classification of the regions can be made to provide additional information to the system. This classification can be considered another type of user input although it would be preferable to use an automatic method such as was described in Section 4.7. Figure 5.7 shows one such categorization for the 75 regions discovered for the Ashcroft image.

This classification is very crude because it has only four categories and two of them are ambiguous (urban/hills and road/mountain). Furthermore, it has inaccuracies due to some incorrectly labelled regions (e.g. some urban regions have been labelled "water" and vice versa). Thus, the categorization does not solve the problem of interpretation; it.is only another type of available information--the "INTERPRET" box in Figure 5.1.

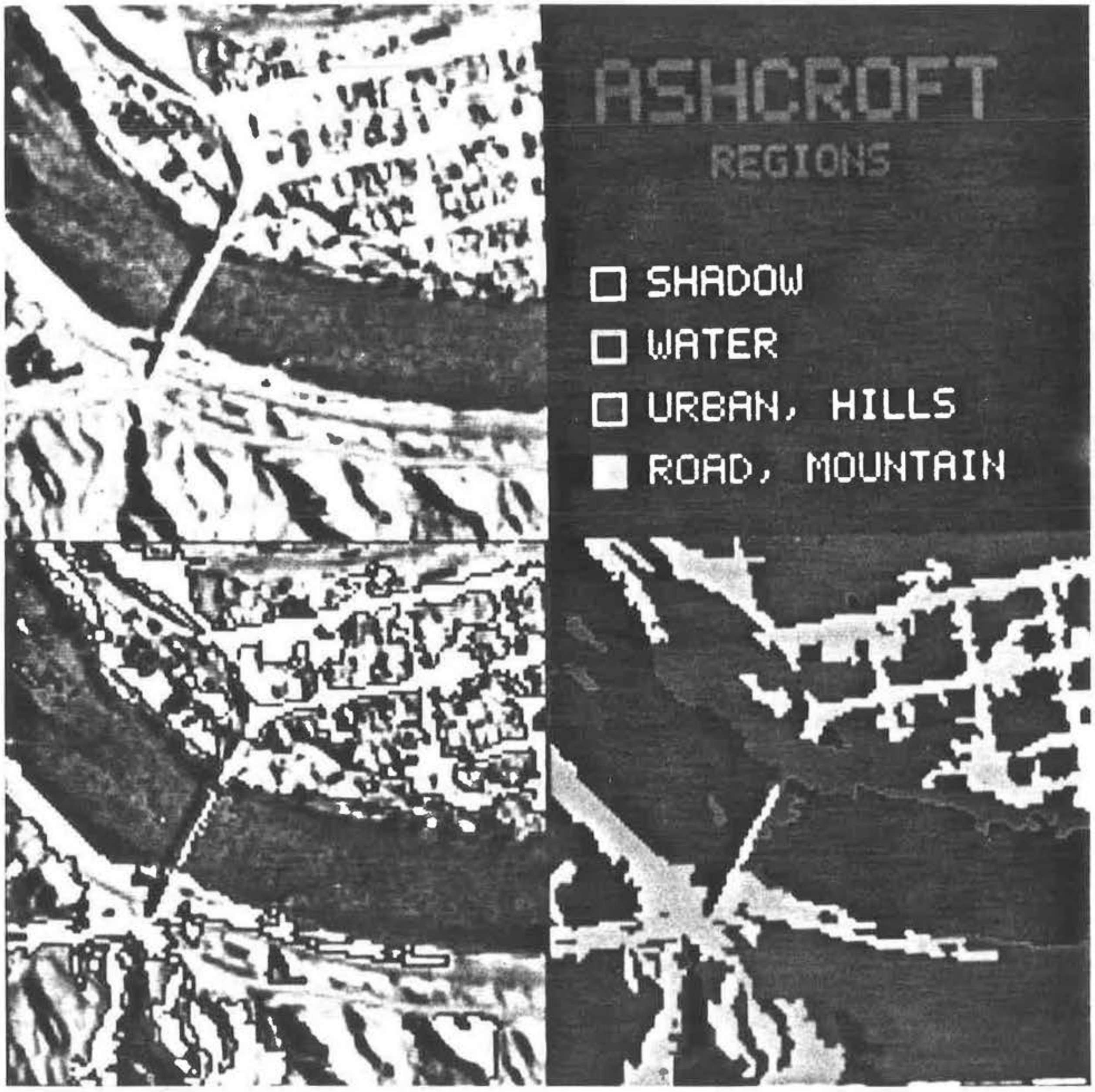


Figure 5.7 Ashcroft: Region Classification

a. (upper left) Ashcroft
b. (lower left) 75 regions
c. (lower right) classified regions

The classification is based solely on the average intensities of the regions. There are several ways that the decision boundaries can be derived: 1) values that were found to work for a different image can be used, 2) the boundaries can be moved on an image display (with function mapping) until one "sees" where a reasonable division point should be, or, 3) as was actually done, a training set of regions and their categories can be used to fix the boundaries for all the regions.

Even though it does not exactly fit the definition in Section 4.5.2, the classification shown in Figure 5.7 corresponds (for each category) to the conservative range [a,b]. The mismatch from the definition arises because there will be some regions in Cn having an average intensity in the range, such as the urban regions classified as water. Nonetheless, since the schemata take the crudeness of the interpretation into account, and require other supportive evidence for instantiation, it is acceptable. In MISSEE, a second series of overlapping ranges is used, corresponding to [a',b'], that ensures that all regions will be included in the proper category range. The interval of acceptance relies on a two step function, such that with no *a priori* information the restricted range, [a,b], is used, while with some reasonable expectation for the categorization, the larger range, [a',b'], is employed.

Even though the classification of regions is crude and inaccurate, it represents a realistic result that might come from any automatic classifier. Statistical pattern recognition techniques are insufficient for interpreting complicated images. More knowledge is required.

## 5.3.2. The Sketch Map

Sketch maps are analyzed by Mapsee2 (Havens and Mackworth, 1980), which like MAIDS is written in Maya. Input is a series of moves/draws (or plots/gotos) which trace the curves of the sketch. Output is a decomposition hierarchy of instances of geographic objects ranging from points, lines, and chains to roads, rivers, mountains, mountain ranges, road systems, geosystems, and the world. Specifics can be found in Appendix D.

Figure 5.8 shows a sketch map of the Ashcroft area. It is a simple task to draw this over an image displayed on a suitable device (in this case a Comtal Vision I). Figure 5.9 shows this combination. The points in the sketch map can then be grouped into lines and made available to Mapsee2.

Mapsee2 results that are made available to MISSEE are a number of Maya schemata (cf. Section 3.3). Points in the sketch curves are joined into links and then chains. Chains are generalized (Appendix B) into a tree of lines representing a binary breakdown of the curves into linear segments.

Chains are generally ambiguous cues for objects such as roads, rivers, and mountains[2]. The shapes of the chains and

[2]The exception are closed chains which are unambiguous cues for shores and blobs which map to towns.

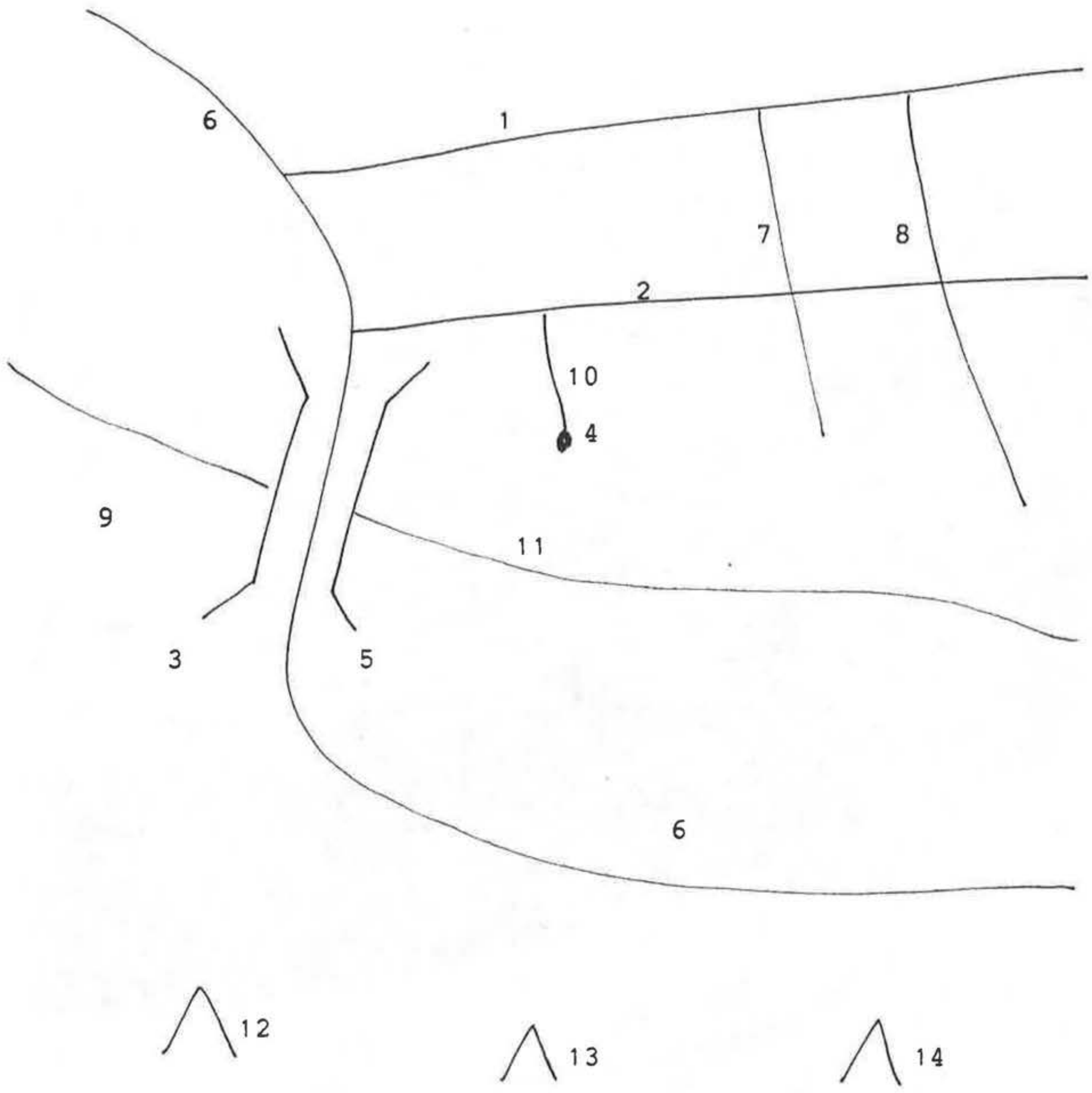


Figure 5.8 A Sketch Map of Ashcroft

their spatial relationships are used to build aggregate structures, such as road systems, river systems, mountain ranges, and the like and to remove some of the labels for the chains. After all the chains have been examined, though, some of the labels will remain ambiguous, which results in a possibly large number

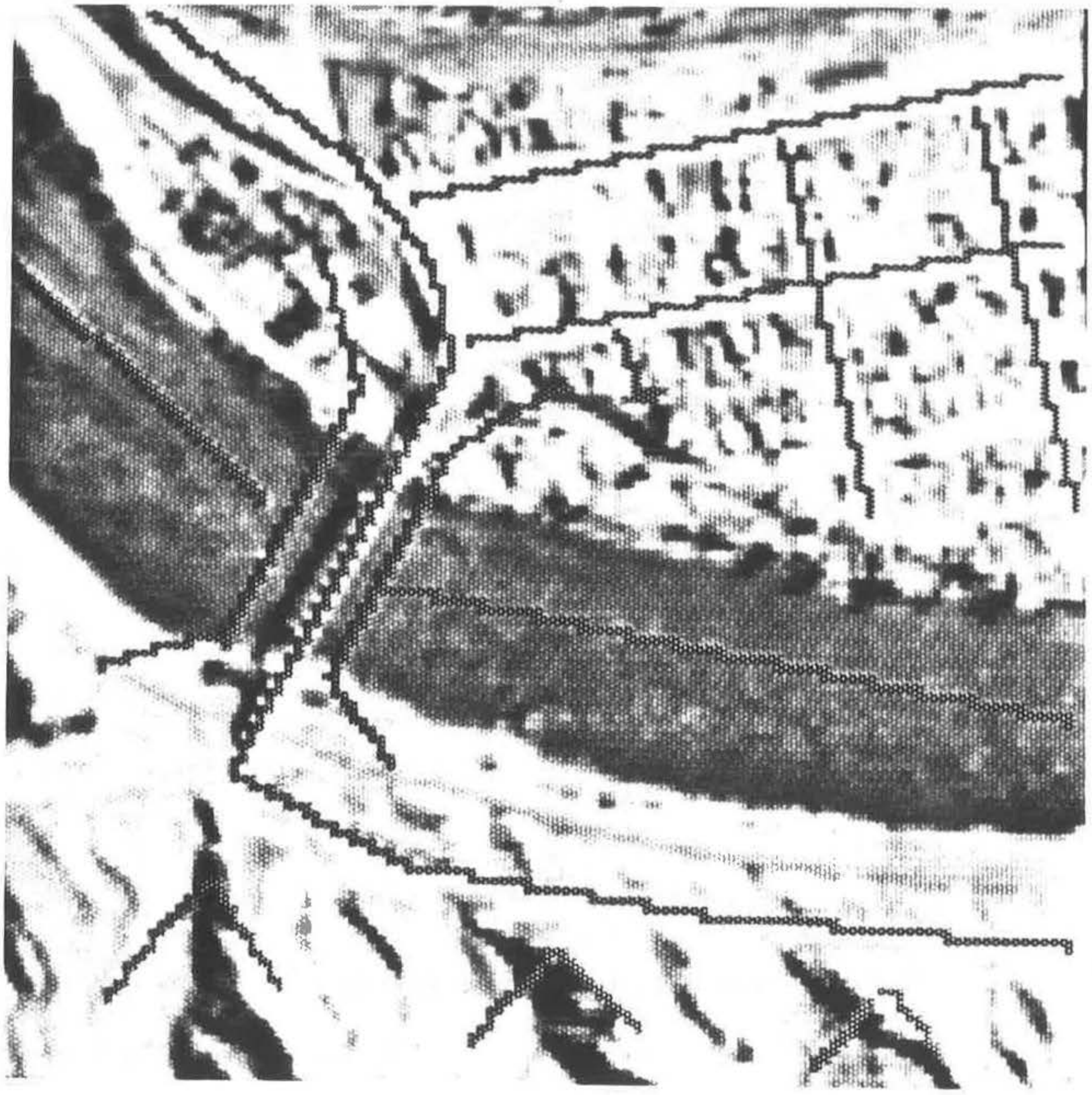


Figure 5.9 Ashcroft with a Superimposed Sketch Map

of hypothetical instances.

Table 5.1 shows all the labels for the chains in the Ashcroft sketch map. The chain numbers correspond to the values in Figure 5.8. The "interpretation" column lists the intended interpretation first. These instances of geographic objects along with the corresponding chains, lines, and points which store their locations are the major source of sketch map information used by MISSEE.

The instances are embedded in a decomposition hierarchy that groups related objects. Figure 5.10 shows the hierarchy built from the "intended" interpretations only. Higher level model knowledge about spatial connectivity could be used by MISSEE. Unfortunately, when MISSEE was developed, the then current version of Mapsee2 did not collect enough of this information.

| CHAIN | LINKS | LINES | INTERPRETATIONS |
|---|---|---|---|
| 1 | 1 - 31 | 1 - 73 | *road-6 *river-6 |
| 2 | 32 - 56 | 74 146 | *road-4 *river-4 |
| 3 | 57 - 101 | 147 239 | *bridge-1 (first side) |
| 4 | 102 - 104 | 240 244 | *town-1 |
| 5 | 105 - 152 | 245 339 | *bridge-1 (second side) |
| 6 | 153 - 292 | 340 648 | *road-5 *river-5 |
| 7 | 293 - 313 | 649 701 | *road-7 *river-7 |
| 8 | 314 - 332 | 702 752 | *road-8 *river-8 |
| 9 | 333 - 406 | 753 899 | *river-1 *road-1 |
| 10 | 407 - 411 | 900 908 | *road-3 *river-3 |
| 11 | 412 - 439 | 909 991 | *river-2 *road-2 |
| 12 | 440 - 475 | 992 1060 | *mountain-1 *road-9 *river-9 |
| 13 | 476 - 508 | 1061 1125 | *mountain-2 *road-10 *river-10 |
| 14 | 509 - 530 | 1126 1168 | *mountain-3 *road-11 *river-11 |

Table 5.1 Interpretations of the Ashcroft Chains

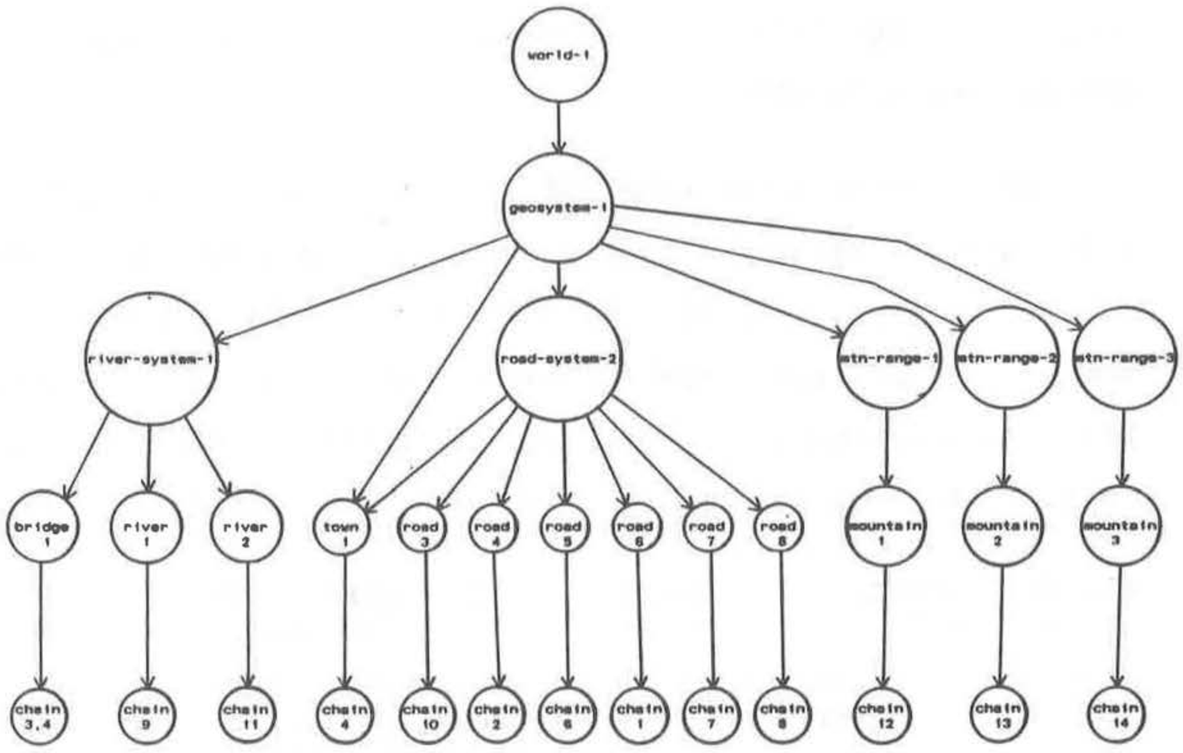


Figure 5.10 A Mapsee2 Decomposition Hierarchy of Ashcroft Sketch Map Instances

In particular, the connection between two roads or a road and a town were discovered only if the end of the road chain is near the other chain. Also, if a road went over a bridge or a river went under a bridge, that fact was not recorded. Use of this type of information would become very important if one were to relax the condition that the sketch map be registered to the image.

Maya schemata are less structured than MAIDS schemata. However, MAIDS schemata are upwards compatible with those used by Mapsee2. The schema-specific routines that use information from sketch maps use only the primitive form of the retrieval functions.

Digitized images contain information that is different in kind from that in sketch maps. Some information derived from images can be useful in interpreting sketch maps. For instance, a closed curve in a sketch map can be unambiguously labelled a shore. However, this gives no indication whether it is a lake (water inside) or an island (water outside). If the location of the shoreline can guide the search for the appropriate features in the intensity image, then a simple pixel classification scheme, as was described in the last section, will generally suffice to determine which region corresponds to water and which to land. While Mapsee2 and MISSEE do not cooperate in this way at this time, the potential exists for the interpretation of each knowledge source to provide useful information to aid the interpretation of the other source.

## 5.3.3. The User

The user can influence several aspects of the interpretation process. He/She can provide certain types of information to the system and can determine when and how it is to be used. In addition, he/she can modify the interactive environment to allow differing amounts of information to be displayed.

The user's means of providing information and influencing the interpretation is via the priority queue used in the cycle of perception. The exact means by which this is accomplished is described in Section 5.6. Unfortunately, the interface is rather "unnatural" in that the user must know the format of internal messages and, while graphic output is used, there is no means for receiving graphic input--other than the sketch map. This is due to the lack of interactive hardware on our current system.

Section 4.4.1 described the three types of input available to a user. Global input pertains to the image as a whole and includes the setting of global values such as the scale of the image (cf. Section 4.6). In addition, global variables are used to determine the output available to the user. Depending on what hardware is accessible, the user can direct images that illustrate the instantiation process to the desired device. This feature is represented by the "images" cloud in Figure 5.1. Also, all the output statements that are directed to the protocol contain a number which must be less than a global printlevel variable before they are actually printed. Hence, the output can be very selective or very complete depending on the needs of the user.

In MISSEE, the global information, "there is an X in the image" is equivalent to the priorities input statement, "find an X in the image". General requests to find specific objects, such as a road or a river system, can be made by adding the appropriate message to the priority queue. In addition, one can change the priorities of the interpretation by rearranging the elements in the queue. By altering the parameters of the message, one can change the context for interpretation and introduce more local information to the system.

Local input is concerned with what specific features the user sees in the image and where they are. Along with an object (e.g. road), the context of a corresponding item in the sketch map (e.g. *road-7) provides useful information. Additionally, the location of the object can be specified as either a region in the intensity image or a position. While this gives the user the ability to specify everything that is important in the image, but still allowing MISSEE to do useful bookkeeping chores such as calculating the length of a river, it should be reiterated that, in a robust system, it is not necessary.

## <u>5.4. Instantiating Schemata</u>

MISSEE interprets images of small urban areas. Figure 5.11 shows the models used in their decomposition and specialization hierarchies. While this is only a small portion of the domain exhibited in Figure 4.4, it is sufficient to classify the scenes that will be used. This section describes how MISSEE instantiates the schemata in this graph from the information found in particular images and sketch maps.

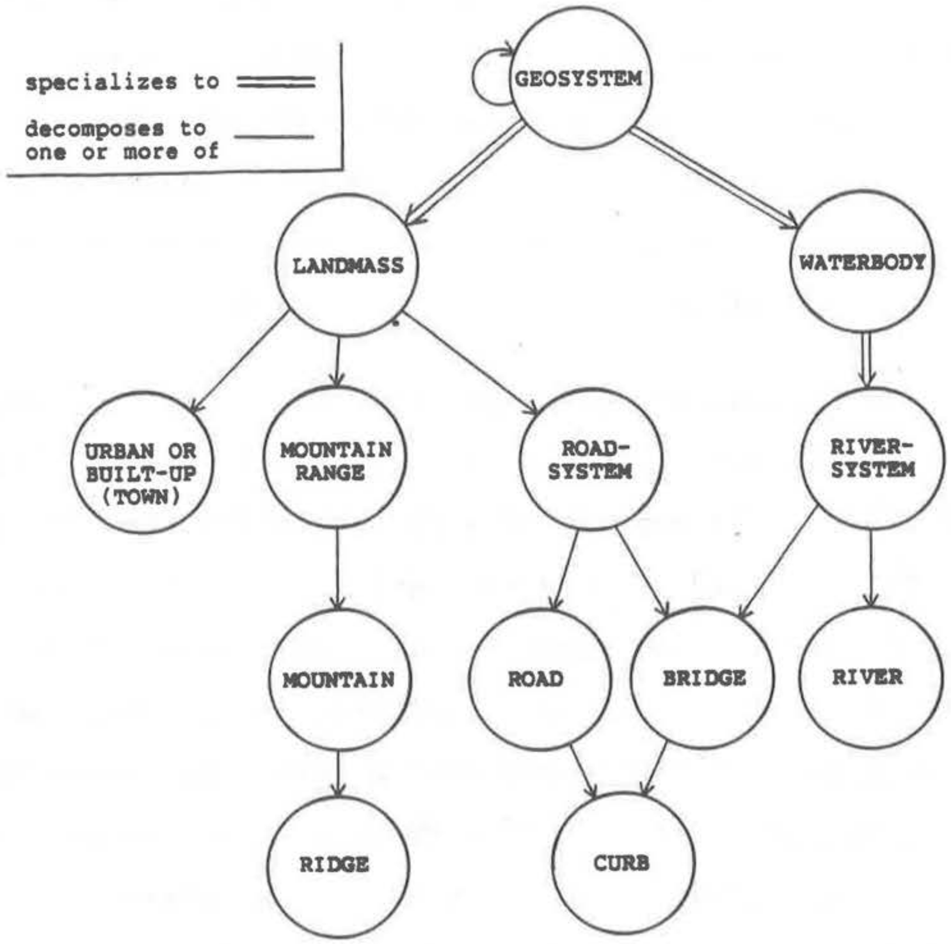


Figure 5.11 The Generic Objects in MISSEE

### 5.4.1. Instantiation Directly from the Information Sources

The five "bottom level" schemata--town, mountain, road, bridge, and river--must directly interact with one or two IS's when they are instantiated bottom-up[3]. This is generally where the interface occurs when processing first starts. The particular object that is first selected may be chosen by the user or from general model knowledge. For example, the landmass schema "knows" that road systems are easier to find than mountain ranges. The location in the image may again be specified by the user or by model knowledge. For example, the largest region in the image can be examined first.

Figure 5.12 is the generic MISSEE schema for a bridge. Instantiation of this schema involves filling in its slots by finding the appropriate features in the IS's. There are two bottom-up procedures attached to bridge: one that requires a sketch map registered to the image and one that deals only with the digitized image. The following two subsections illustrate (in pseudo-Lisp) how these procedures work (Glicksman, 1982).

#### 5.4.1.1. Instantiating %bridge-1 with the Aid of a Sketch Map

This section consists of brief excerpts of "pseudo-code" (preceded by "|") and explanations of what is accomplished. Responses from the program are indented and preceded by ">".

[3]Ridges and curbs are created as part of this process, they are not instantiated independently.

```
sketchmapitem:     value: nil  %confidence: nil

                   %if-added:     (prog nil
                    (printlb "value added to sketch
                             map item = " %val))
                   %if-removed:   (prog nil
                    (printlb "value removed from sketch
                             map item = " %val))
                   %if-modified:  (prog nil
                    (printlb "value modified in sketch
                             map item = " %val))

orderlist:         value: nil  %confidence: nil
neighbourregions:  value: nil  %confidence: nil
shadowregions:     value: nil  %confidence: nil
roadregions:       value: nil  %confidence: nil

a-part-of->        (%road-system %river-system)
decomposes-to->    (%curb)
instances->        nil
neighbours->       nil

confidence:        nil
conf-alg:
  (prog (val lst)
     (setq val (cond ((sgetv %name 'sketchmapitem 'n) 50.0)
                     (t 25.0)))
     (setq lst (sgeta %name 'decomposes-to))
     (cond ((null lst)(return val))
           ((atom lst)(return (plus val
                                    (times 0.2 (sgetc lst)))))
           (t (return
                (plus val
                      (times 0.2
                             (quotient
                                (apply 'plus
                                   (mapcar
                                      '(lambda (n) (sgetc n))
                                      lst))
                                (length lst))))))))
```

Figure 5.12 The Stereotype %bridge Schema

Schemata manipulation functions will be underlined. If *bridge-1 has been selected from the sketch map analysis results (cf. Figure 3.6) to aid in the instantiation of a bridge, then the variable SKETCHMAPITEM will be bound to it when the procedure is entered.

The basis of the routine is the following: The routine searches for regions in the appropriate area that can be interpreted as road or shadow. Then edges are found in the same area which can define the sides of the bridge and possibly a shadow.

```
1.| if (SKETCHMAPSCHEMA =
  |      (sgetv '%bridge 'sketchmapitem 'yes 'one '(instances))
  |      (return 'alreadyexists))
```

This is a check to determine whether this schema has been examined before. Sgetv searches from %bridge down the "instances" LINKs looking for one instance schema containing SKETCHMAPSCHEMA as its sketch map item.

```
2.| Inst = (snewi '%bridge)
  |      > create a new bridge instance: %bridge-1
  |
  | p1 = (car (sgeta SKETCHMAPSCHEMA 'side1-desc))
  | p2 = (car (sgeta SKETCHMAPSCHEMA 'side2-desc))
  |      > p1 = (57 . 33)
  |      > p2 = (69 . 62)
```

Since Mapsee2 schemata are undifferentiated, sgeta returns the value of slots. p1 and p2 are the locations of the midpoints of the bridge sides.

```
3.| reglist = (pointstrips p1 p2 %regmatfile
  |               (quotient 15.0
  |                          (sgetv '%scale 'feetperpixel 'no
  |                                 'valueordefault 'one)))
  |     > regions list = (130 1186 1629 9 1750 2018)
```

Pointstrips searches for all the regions (generated from a region-merging algorithm) in %regmatfile that are enclosed in a rectangular strip whose corners are 15 feet from p1 and p2.

```
4.| orderlist = (ordinterp reglist Inst)
  |      > order list = (Other Shadow Bridge Bridge Other)
```

Orderinterp interprets the regions and determines if their interpretations are consistent with regions around and over a bridge. Interpretations include ROAD, WATER, URBAN, SHADOW, MOUNTAIN, and HILLS. The appropriate regions are added to the "%bridge-1" schema in the slots "shadowregions", "roadregions", and "neighbourregions". The order list shows the order of regions from p1 to p2. In this case they are Other, Shadow, and Bridge.

```
5.| (sputv Inst 'orderlist orderlist)
```

Put the value of orderlist in the VALUEd type of the same name in Inst.

```
6.| (sputc Inst 'orderlist 100)
```

Put the confidence of orderlist at 100 (the maximum).

```
7.| edgelist = (pointstrips p1 p2 %edgematfile
  |                 (quotient 15.0
  |                           (sgetv '%scale 'feetperpixel 'no
  |                                  'valueordefault 'one)))
  |      > edges list = (88 11 205 206 241 242)
```

Find all the edge segments in the same rectangular strip that was searched for regions. Edge segments come from groups of zero-crossings of the Laplacian of the Gaussian applied to the image.

```
8.| if (null edglist)
  |      then (sfail-model Inst)
```

If there are no edge segments then this model can not be correct. So, remove it and any of its descendants from the graph.

```
9.| if (notwithinrange edgeangle bridgeorientation)
  |      then (remove edge edglist)
  |      > edges list = (88 11 242)
```

Remove all edge segments that are not similar to the orientation of the bridge in the sketch map, which is 67 degrees.

```
10.| forall i in edgelist
   |     (addto edgelist (lineneighbours i -1.0 35 %edgefile))
   |       > edges list = ((88 89)(10 11 12 13 14 15) (242 243))
```

Expand all the edge segments in edgelist to include segments out of the rectangular strip that are connected to and in a similar orientation as existing edge segments.

```
11.| (foreach group of edges i in edgelist
   |      (linestats i))
```

Linestats calculates the length of the edge segments, the average contrast across the edge, and the maximum contrast.

```
12.| (matchregionstoedges orderlist edgelist Inst)
```

Match the regions from regionlist with the edges in edgelist. One result of this is the creation of new instances of the %curb schema. Each %curb schema contains the data for one edge of the bridge. A special type of %curb is reserved for shadows. After this routine has executed, the schemata shown in Figure 5.13 will exist. The confidence algorithm is executed after the last value slot is filled by a call to sputc with the appropriate flag set.

```
13.| riverreg = (sgetv '%river 'regions 'yes
   |                   'valueonly 'all '(instances))
   |       > riverreg = ((%river-1 130 1058 943 874)
   |                     (%river-2 1750 2018 2095))
   |
   | forall regions r in neighbourregions
   |     if r is in riverreg
   |     then (saddl Inst 'neighbours River))
```

```
avgstrength: value: 3.5       %confidence   75
maxstrength: value: 5         %confidence   75
length:      value: 20        %confidence  100
angles:      value: (248 214) %confidence  (80)
edgesegs:    value: (88 89)   %confidence  100
type:        value: shadow    %confidence  100

a-part-of->     %bridge-1

conf-alg: (prog nil (return
           (quotient
              (apply 'add (cons (sgetc %name 'length)
                                (cons (sgetc %name 'maxstrength)
                                (sgetc %name 'angles))))
              (add 2 (length (sgetc %name 'angles))))))

confidence:     85

                5.13a %curb-1
    ######################################

avgstrength: value: 96.33     %confidence  100
maxstrength: value: 134       %confidence  100
length:      value: 54        %confidence  100
angles:      value: (90 57 79 45 90 27)
                              %confidence  (80)
edgesegs:    value: (10 11 12 13 14 15)
                              %confidence  100
type:        value: road      %confidence  100

a-part-of->     %bridge-1

confidence:     93

                5.13b %curb-2
    ######################################

avgstrength: value: 18.0      %confidence  100
maxstrength: value: 22        %confidence  100
length:      value: 20        %confidence  100
angles:      value: (228 270) %confidence (100)
edgesegs:    value: (242 243) %confidence  100
type:        value: road      %confidence  100

a-part-of->      %bridge-1

confidence:      93

                 5.13c %curb-3
```

Figure 5.13 Three %curb Instances

```
        else (addtoqueue 'td '%river (sgetc Inst)
                              (list 'region r)))
             (adddemon (list 'demonaddlink '%bridge 'regions
                              r Inst 'neighbours))
         > neighbours = (%river-1 %river-2)
```

Search through all the existing river schemata for the regions they contain. If they match the neighbouring regions of the bridge, add links to the rivers from the bridge and vice versa. If not, send a message to the %river schema to try to instantiate it with the appropriate regions. If the %river schema is subsequently instantiated, a demon will establish the neighbours link between it and %bridge-1.

```
14.  roadreg = (sgetv '%road 'regions 'yes
                         'valueonly 'all '(instances))
          > roadreg = nil

     forall regions r in roadregions
         if r is in roadreg
         then (saddl Inst 'neighbours Road))
         else (addtoqueue 'td '%road (sgetc Inst)
                               (list 'region r)))
              (adddemon (list 'demonaddlink '%bridge 'regions
                               r Inst 'neighbours))
          > message added to QUEUE: (td %road 68 (region 1629))
          > DEMON initiated: demonaddlink
```

Repeat the actions of 13 for the road that passes over the bridge.

```
15.  (sputv Inst 'sketchmapitem SKETCHMAPSCHEMA)
     (sputc Inst 'sketchmapitem 100 t)
          > value added to sketchmapitem = *bridge-1
```

Instantiation has succeeded so add SKETCHMAPSCHEMA as the sketch map item for this schema. Set its confidence to the maximum and propagate that fact. Figure 5.14 displays the schema for %bridge-1 at this stage.

```
16. (addtoqueue 'bu '%river-system (plus 5 (sgetc Inst))
                                    (list Inst))
    (addtoqueue 'bu '%road-system (plus 5 (sgetc Inst))
sketchmapitem:       value: *bridge-1
                     %confidence:        100

orderlist:           value: (Other Shadow Bridge Bridge Other)
                     %confidence:        100
neighbourregions:    value: (2018 1750 9 130)
                     %confidence:        100
shadowregions:       value: (1186)
                     %confidence:        100
roadregions:         value: (1629)
                     %confidence:        100

a-part-of->        nil
decomposes-to->    (%curb-1 %curb-2 %curb-3)
neighbours->       (%river-1 %river-2)

confidence:        68
```

Figure 5.14 Instance %bridge-1

```
                                  (list Inst))
        > messages added to QUEUE:
                     (bu %river-system 73 (%bridge-1))
                     (bu %road-system 73 (%bridge-1))
```

Send messages to the two higher level schemata that bridges are part of. %bridge-1 will either find existing road and river system instances to become part of, or will cause them to be created.

Figures 5.15 and 5.16 show graphically the regions and edges (respectively) that were used to instantiate %bridge-1.

### 5.4.1.2. Instantiating %bridge-2 from the Intensity Image

Like the last section, this section contains pseudo-code and explanations of how one can instantiate a bridge from the intensity image alone. The argument the procedure enters with,

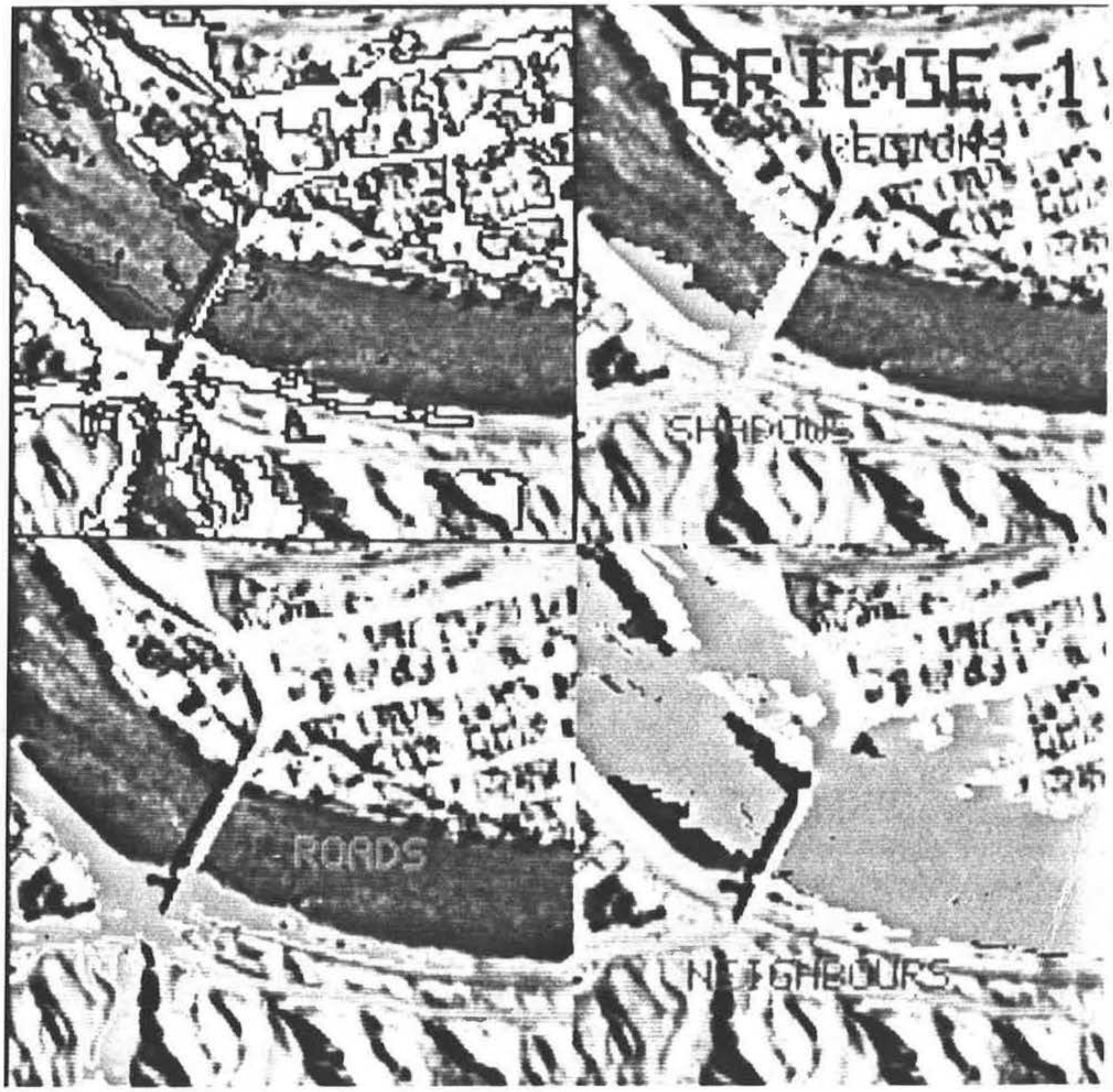


Figure 5.15 Ashcroft: %bridge-1 Regions

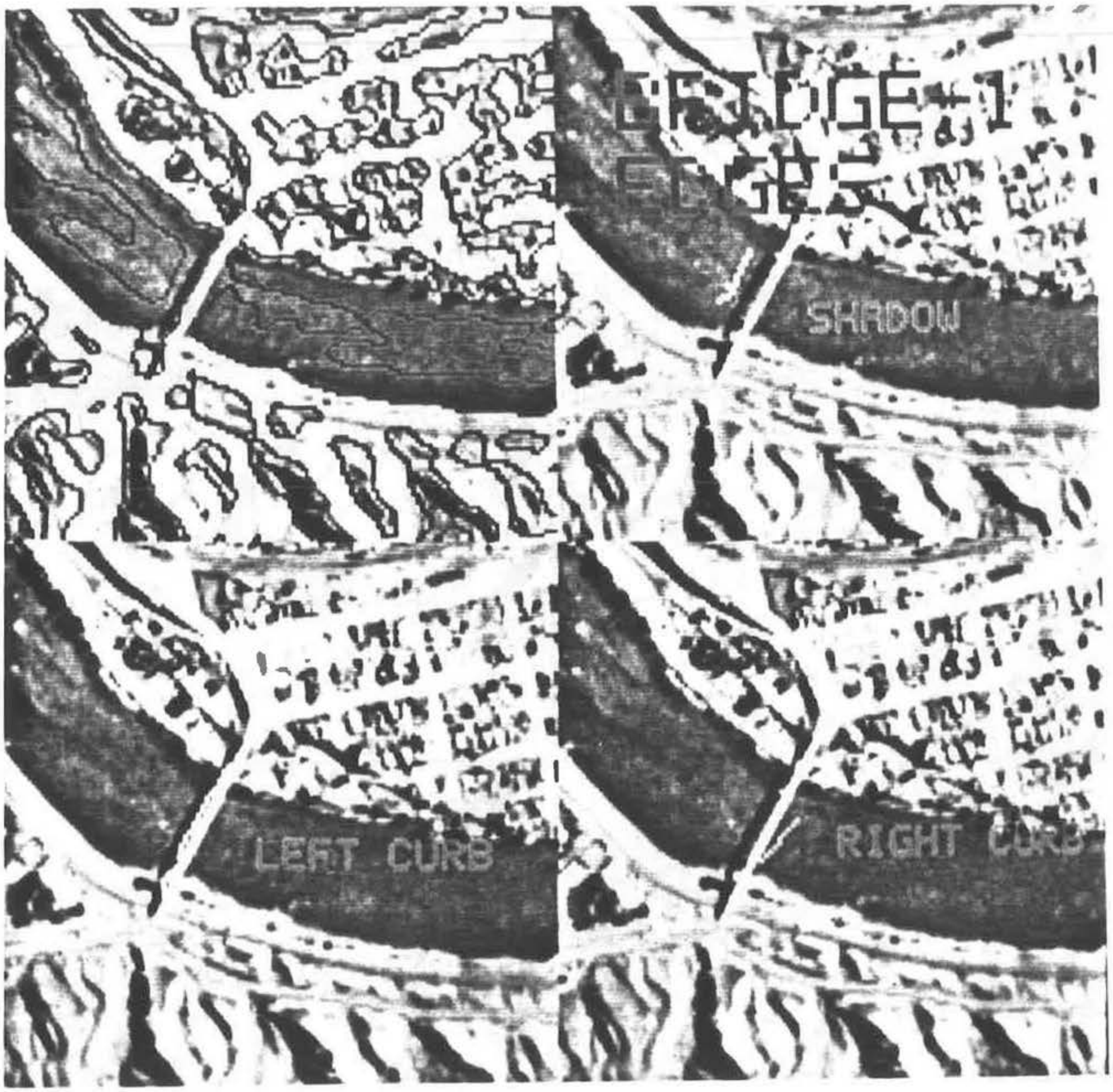


Figure 5.16 Ashcroft: %bridge-1 Edge Segments (Curbs)

if it exists, is a number denoting a region to be searched for the bridge.

The routine finds regions that can have the interpretation WATER surrounding a region with the interpretation ROAD. The positions midway along the common boundaries are used to fix the edges of the possible bridge.

```
1. | if #args = 1
   |    then reg = (arg 1)
   |         if (or (not (memq 'ROAD (regioninterpret reg)))
   |                (regioncheck reg '%bridge))
   |            then (return nil)
   |            else go to 3.
```

Whenever there is an argument, it is a specific region in the image to be used. One of its possible interpretations (cf. Section 5.3.1.2.1) must be ROAD and it should not have been previously used to attempt bridge instantiation.

```
2. | minsize = (quotient
   |               (times 50 50)
   |               (square (sgetv '%scale 'feetperpixel 'no
   |                              'valueordefault 'one)))
   |
   | reg = (nextlargestregion '%bridgearea minsize 'ROAD)
   | if (regioncheck reg '%bridge))
   |    then go to 2.
   | if (null reg)
   |    then (return nil)
   |      > region = 18     area = 1695
```

If there is no region specified, find the largest region in the image that can have ROAD as one of its interpretations and has not been tried before. Stop searching when the area of the region is less than 2500 square feet.

```
3. | neighb = (neighbourregions reg)
   |      > neighbours = (19 9 1359 738)
```

Find all the regions that share a common boundary with reg.

```
4.| forall regions n1 in neighb
  |   if (memq 'WATER (regioninterpret n1))
  |      then n2 = (moveacross (commonboundary reg n1))
  |           if (memq 'WATER (regioninterpret n2))
  |              then go to 5.
  | if (#args = 1)
  |    then (return nil)
  |    else go to 2.
  |
  |       > neighbour1 = 1359
  |       > no proper neighbour pairs for region 18
  | from 2. > region = 1629       area = 1330
  | from 3. > neighbours = (1186 130 9 2131 1750 2201 2293
  |                         and 13 others)
  | from 4. > neighbour1 = 130
  |         > neighbour1 = 1750
  |         > neighbour2 = 130    shadow region = 1186
```

Check each region in neighb to see if it can be interpreted as WATER. If so, find the midpoint of the common boundary between the two regions. Then move back across reg (1 pixel at a time) in the direction perpendicular to the boundary of n1 and reg. If the first new region that is reached can be interpreted as SHADOW, store this fact and keep going in the same direction. If the new region can be interpreted as WATER, then a good pair of bounding regions has been found. If no suitable pair of neighbours is found in neighb, then if reg was given as an argument, give up, else go back and reset reg to the next largest region.

```
5.| Inst = (snewi '%bridge)
  |      > create of new bridge instance: %bridge-2
```

Only now do we have enough confidence and information to create an instance for the bridge. Processing continues much as it did in the last section when the sketch map was used. p1, p2, and orderlist are determined from the results of the "commonboundary" and "moveacross" functions. From step 5 on, instantiation continues almost identically.

For this particular image, the results from both procedures are similar. Note, however, that the second routine is both more expensive computationally and more prone to missing the features necessary to establish the existence of the bridge. Figure 5.17 shows the final schema for %bridge-2. The curb schemata are identical to those in Figure 5.13.

## 5.4.2. Building up the Hierarchies

In Section 5.4.1, it was shown how two of the four relationships among schemata are formed. Instances are created when there is sufficient evidence to support the hypothetical existence of a geographic object. New evidence, or a lack

```
sketchmapitem:      value: nil
                    %confidence:        nil

orderlist:          value: (Other Shadow Bridge Bridge Other)
                    %confidence:        100
neighbourregions:   value: (1750 130)
                    %confidence:        100
shadowregions:      value: (1186)
                    %confidence:        100
roadregions:        value: (1629)
                    %confidence:        100

a-part-of->        nil
decomposes-to->    (%curb-4 %curb-5 %curb-6)
neighbours->       (%river-1 %river-2)

confidence:        43
```

Figure 5.17 Instance %bridge-2

thereof, may cause the instance to be destroyed. LINKs to the generic objects are implicitly created and destroyed with the instance. Instantiation provides information that is used to determine the neighbour relationships. When bridges are instantiated, the region going over the bridge belongs to a road and at least two regions beside it belong to rivers. In Section 5.4.1.1 it was assumed that two rivers had already been instantiated and because they shared regions in common with %bridge-1, the neighbour LINK was formed (step 13). It was assumed that the road had not yet been established so no LINK could be formed (step 14). However, if and when the road associated with that region was instantiated, a demon would add the LINKs between the two schemata.

Also present in Section 5.4.1.1 were the messages that would cause the specialization and decomposition hierarchies to be built up (step 16). When higher level schemata such as road system and river system are entered from below, they try to fit new instances into the existing semantic network or build on new additions to the network itself. This communication requires schemata to have knowledge of what is above them in the hierarchies, but if demons were used (see Section 5.6) it would be possible for that knowledge to exist in the higher level schemata alone.

A straightforward algorithm for building up the hierarchies is shown in Figure 5.18. There are three basic possibilities. 1) If the new instance is not spatially near any existing

HIGH--the type of high level schema being joined
NEW --the new instance to be put into the hierarchies

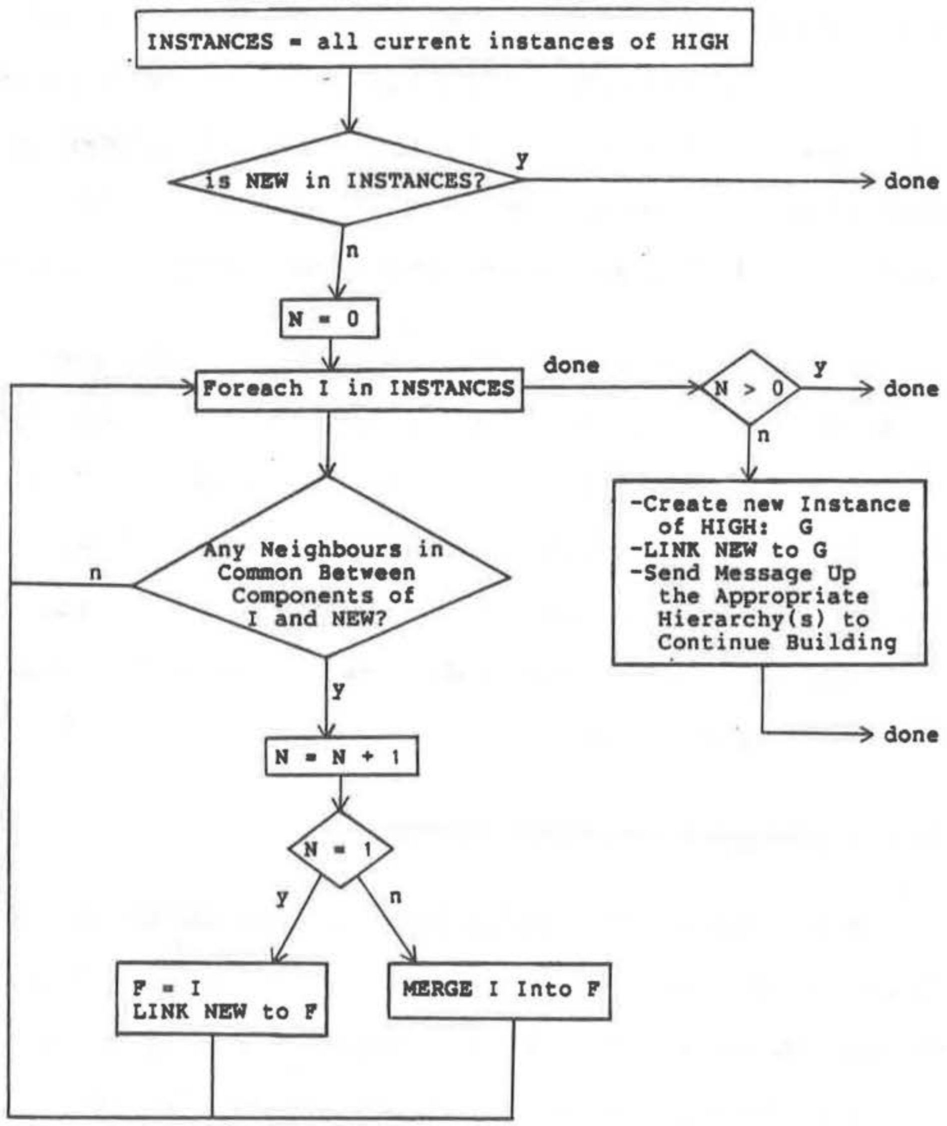

INSTANCES = all current instances of HIGH
is NEW in INSTANCES?
y
done
n
N = 0
Foreach I in INSTANCES
done
N > 0
y
done
n
-Create new Instance of HIGH: G
-LINK NEW to G
-Send Message Up the Appropriate Hierarchy(s) to Continue Building
done
Any Neighbours in Common Between Components of I and NEW?
n
y
N = N + 1
N = 1
y
n
F = I
LINK NEW to F
MERGE I Into F

Figure 5.18 Building up the Semantic Network

elements belonging to a higher level schema it creates a new instance of that schema and sends another message up the appropriate hierarchies. 2) If the new instance is near an element of one higher level node it becomes a new descendant of that node. 3) If it shares neighbours with more than one node then the higher level instances are all merged into one new schema which also becomes the father of the new instance. This algorithm is generally applicable and requires only that each higher level instance provide the type of LINKs it wishes to establish with its ancestors and descendants in the graph.

An example of a semantic network of instances is shown in Figure 5.19. The generic objects and the AIO links have not been drawn since it is clear from the names used for instances what the correspondence would be. The other three types of LINKs are all indicated. This network would result if all of the "intended" sketch map entities in Ashcroft were instantiated from the intensity image.

## 5.4.3. Top-Down Attached Procedures

In the last two subsections it was shown how bottom-up procedures are used to instantiate schemata and to build up the semantic network. Top-down procedures are used to direct attention to schemata whose instantiation would most profitably advance the interpretation.

Section 4.5.1 described two types of top-down control. The first type applies model knowledge in the absence of other

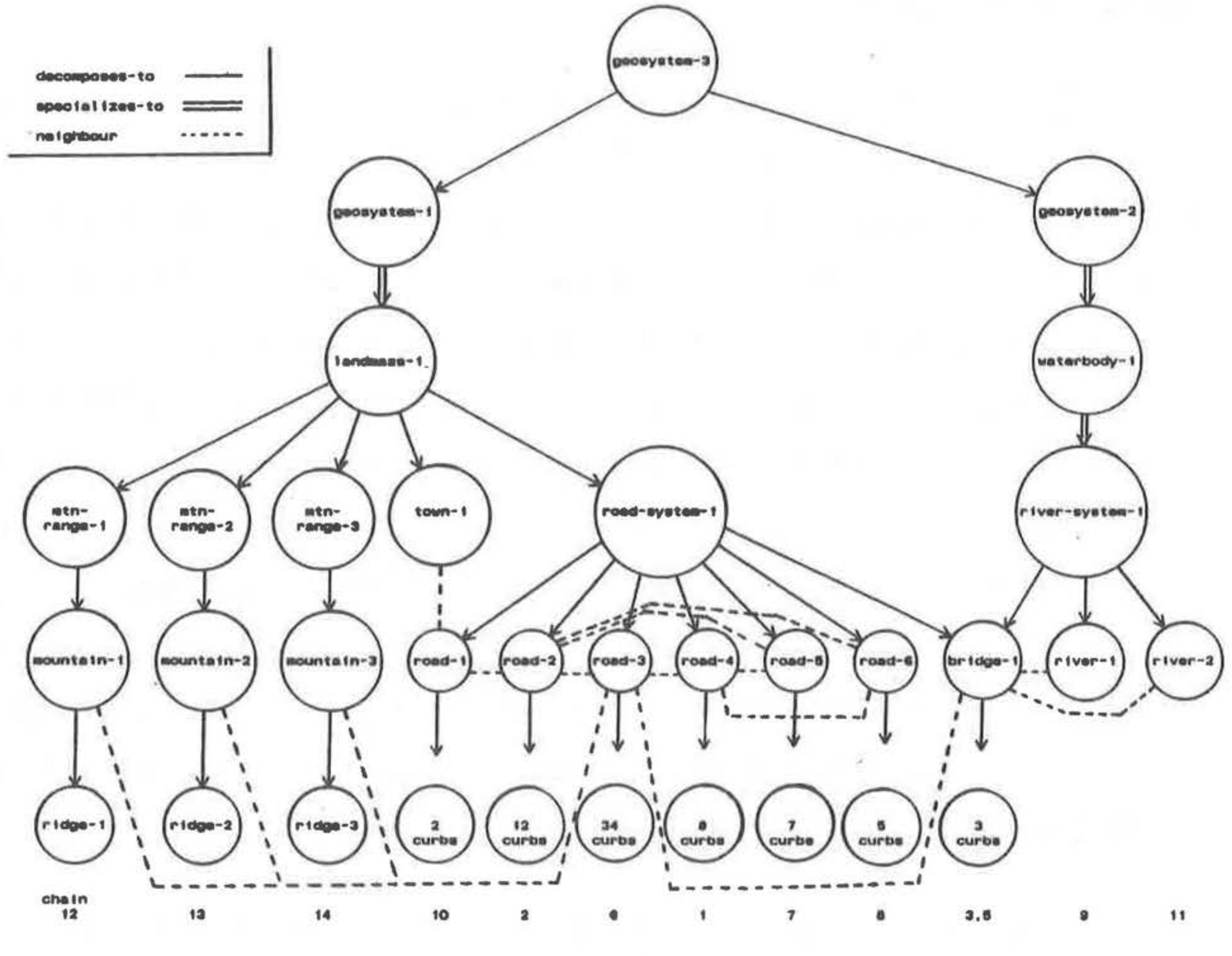


Figure 5.19 An Ashcroft Instance Hierarchy

information. For example, if the user requests, "Find a geosystem", then the geosystem top-down procedure will decide whether to search for a landmass or a waterbody (its two specializations) based on a priori knowledge of which is easier and more reliably established.

All of the higher level schemata contain knowledge of this sort that moves control down the hierarchies. While it would be possible to jump several levels, which would be more efficient, this has not been done because of modularity. Consequently, each schema only needs to know about those schemata it connects to directly. The five bottom level schemata (road, mountain, etc.) have the knowledge that they are at the bottom of the hierarchies and their top-down procedures give control to the bottom-up routines. In the absence of other information, the top-down procedures also try to find a sketch map item or a region that the appropriate bottom-up procedure can be applied to. Note that these top-down routines only redirect control and do not actually instantiate their schemata.

The second type of top-down control takes place when the current context is available to combine with model knowledge. Thus, if the user again requests, "find a geosystem", and there already is an instantiated waterbody, the geosystem might decide to instantiate a landmass. So, the schemata still direct control down the hierarchies, but do it in a more informed manner.

It is also possible for top-down routines to instantiate schemata directly. For example, in the case where the only instances in river-systems were two rivers, the river system procedure might search in the area between the rivers, if they were a suitable distance apart, for a bridge separating them. If the supporting features were found, it could instantiate the bridge directly and merge the three objects into one river-system. This type of procedure has not been implemented, but a similar efficiency gain results from the use of lateral messages.

Section 5.4.1.1 indicates that when %bridge-1 was instantiated, a lateral message was sent in step 14 to the road schema. When schemata are instantiated, some of the knowledge they gain can be used to help instantiate other schemata. In this case, the bridge procedure "knows" that one of its regions should be a road (and others, rivers) and can send a specific message to that schema. The role of the top-down routines for the bottom level schemata then becomes one of making sure the contextual information (in this case the region number) is presented in the proper form to the bottom-up procedure. This is also necessary if the information has come from the user. For example, he/she might specify that an instance from the sketch map analysis use an exact position.

While this system was not designed to be a production system, efficiency questions are still important. One could try to instantiate every region in the image, or worse yet, every point, and every curve in the sketch map with every model. Control would be very simple and similar results would ensue. However, for anything other than trivial images this would prove unfeasible, whereas the methods described herein provide greater flexibility and procedural adequacy.

## 5.5. Cluster Analysis Using Gaussian-Smoothed Histograms

A one-dimensional version of the clustering method described in Section 4.7 has been implemented and used in several places in MISSEE. It has been successfully employed to determine the features that should be used to fill slots during instantiation and to reject the existence of hypothetical schemata.

The method has been applied both in an incremental and a static fashion. The orientation of edges along the sides of a road and of the ridges in mountains are incremental cases. Determining the rectangularity of a region is static.

The ridges in a mountain range at least locally tend to run in the same direction. The orientations of edge segments can be used to check the consistency of that value. Orientation varies over 180 degrees (ignoring the bright/dark direction across the edge) and is cyclic, that is, 0 and 180 degrees represent the same orientation. Several edge segments are discovered at a time, because they are close to a chain in the sketch map, for example, and their orientations are added to the histogram buckets. The algorithm is executed after each new collection of orientations is provided to determine if the new ridge is consistent with the others found so far. Of course, the new information may alter the confidence of the old values (see below) which may result in the confidence value of an existing ridge to be recalculated. This recalculation affects the confidence of mountains, and so on up the hierarchy.

After the orientation histogram is broken into clusters, the following information is returned for each orientation value: the number of samples in its cluster (n), the mean of its cluster (m), and the variance of its cluster (v). Then for each orientation value, a confidence level is calculated. The function used in MISSEE is

$$C(o) = \begin{cases} \dfrac{100n}{N} & \text{if } v = 0 \\ \dfrac{50n}{N} & \text{if } |o - m| > 3v \\ \dfrac{50n}{N} * \left(2 - \dfrac{|o - m|}{3v}\right) & \text{otherwise} \end{cases}$$

where o is the orientation and N is the total number of values. This function responds both to inter and intra-cluster differences. Clusters with many members will have a larger value of n/N so each element will have a higher confidence value. Within clusters, a function varying from .5 to 1.0 raises the confidence values of elements whose feature value is similar to the mean of the cluster. Each confidence value is scaled to range from 0 to 100. This value is then returned as the confidence for the angles in all ridges being considered for inclusion as well as those ridges previously instantiated.

The orientation of line segments that can be part of roads is handled in a manner similar to ridges. This makes the assumptions that the roads are straight (as is usually the case). Roads that curve (such as the road represented by chain 6 in the Ashcroft sketch map) are not included in the clustering[4]. Since roads in urban areas are often found in grid patterns one can also cluster all the straight roads together using orientation modulo 90 degrees. Figure 5.20 shows the results of clustering for %road-4 (corresponding to chain 1 in the sketch map). Table 5.2 shows the confidence values that are calculated from these results. Since edges 139, 385, 383, 392, and 68 are in the minor cluster they have a much lower confidence. Figure 5.21 shows the edge segments that form %road-4.

## 5.5.1. Determining Rectangularity

If one is assuming that roads are laid out in a rectangular grid, then the urban regions (city blocks) that are bounded by roads should be rectangular. Also, small sections of rivers will also appear to be rectangular if the scale is large.

[4]Curvature is easily determined from the generalized line representation--see Appendix B.

```
  σ          12.0 11.0 10.0 9.0 8.0 7.0 6.0 5.0 4.0 3.0 2.0
#clusters      2    2    3   3   4   6   7   7   9  10  11
best σ         *

  cluster rightend numsamples   mean    variance
  ----------------------------------------------
    1         92       22        19.455  639.066
    2        151        5       126.400   98.640

  σ = 12; number of clusters = 2
|--+++++++++++++++++++++++++|-----------++++++|-------- 1st.
|------+++++-----+++++++++++|++++++--------+++|+++++--- 2nd.
|                           |                 |       *
|            *              |                 |       *
|            *              |                 |       *
|   *        *              |            *    |*      *
|  ******  * *     *  *     |    *   **  *    |*    * *
|---------------------------|-----------------|--------
 0 1  2  3  4  5  6  7  8  9| 1  1  1  1  1  1| 1  1  1
   0  0  0  0  0  0  0  0  0| 0  1  2  3  4  5| 6  7  8
                            | 0  0  0  0  0  0| 0  0  0

                               orientation
```

Figure 5.20 Clustering Orientations in Ashcroft %road-4
1st. the result of convolution with the first derivative (positive or negative).
2nd. the result of convolution with the second derivative.

Hough transforms have previously been used to find bounding rectangles (Sloan, 1982). However, Sloan used "dot product space", the extent of the projection of the boundary points, whereas we will use orientation space. In this method, one moves along the boundary points of the region and calculates the orientation of the line joining every pair of points a specific distance (d) apart to form a histogram of the orientations. By varying both $\sigma$ and d, a two-dimensional table can be produced whose elements are the resulting number of clusters. The centre

| ANGLE | EDGE | CURB | CONFIDENCE |
|---|---|---|---|
| 11 | 71 | 7 | 81.301 |
| 14 | 394 | 2 | 81.365 |
| 14 | 69 | 7 | 81.365 |
| 18 | 387 | 1 | 81.450 |
| 22 | 359 | 3 | 81.427 |
| 23 | 59 | 8 | 81.406 |
| 27 | 136 | 6 | 81.321 |
| 37 | 384 | 1 | 81.108 |
| 45 | 382 | 1 | 80.938 |
| 45 | 367 | 5 | 80.938 |
| 45 | 138 | 6 | 80.938 |
| 45 | 70 | 7 | 80.938 |
| 63 | 393 | 2 | 80.556 |
| 72 | 72 | 7 | 80.364 |
| 109 | 139 | 6 | 17.974 |
| 124 | 385 | 1 | 18.443 |
| 127 | 383 | 1 | 18.499 |
| 136 | 392 | 2 | 18.218 |
| 136 | 68 | 7 | 18.218 |
| 154 | 361 | 4 | 78.622 |
| 158 | 364 | 5 | 78.537 |
| 175 | 135 | 6 | 78.176 |
| 180 | 386 | 1 | 78.069 |
| 180 | 366 | 5 | 78.069 |
| 180 | 137 | 6 | 78.069 |
| 180 | 360 | 3 | 78.069 |
| 180 | 58 | 8 | 78.069 |

Table 5.2 Orientations and Confidence Values

of mass of the largest region in the chart where the number of clusters is constant indicates a "good" choice of parameters. If the figure is rectangular, the heuristic should result in two clusters whose means are 90 degrees apart. This is illustrated in Figure 5.22 for a diamond-shaped rectangle. This method works for other polygonal shapes too. For example, a triangle will result in three clusters.

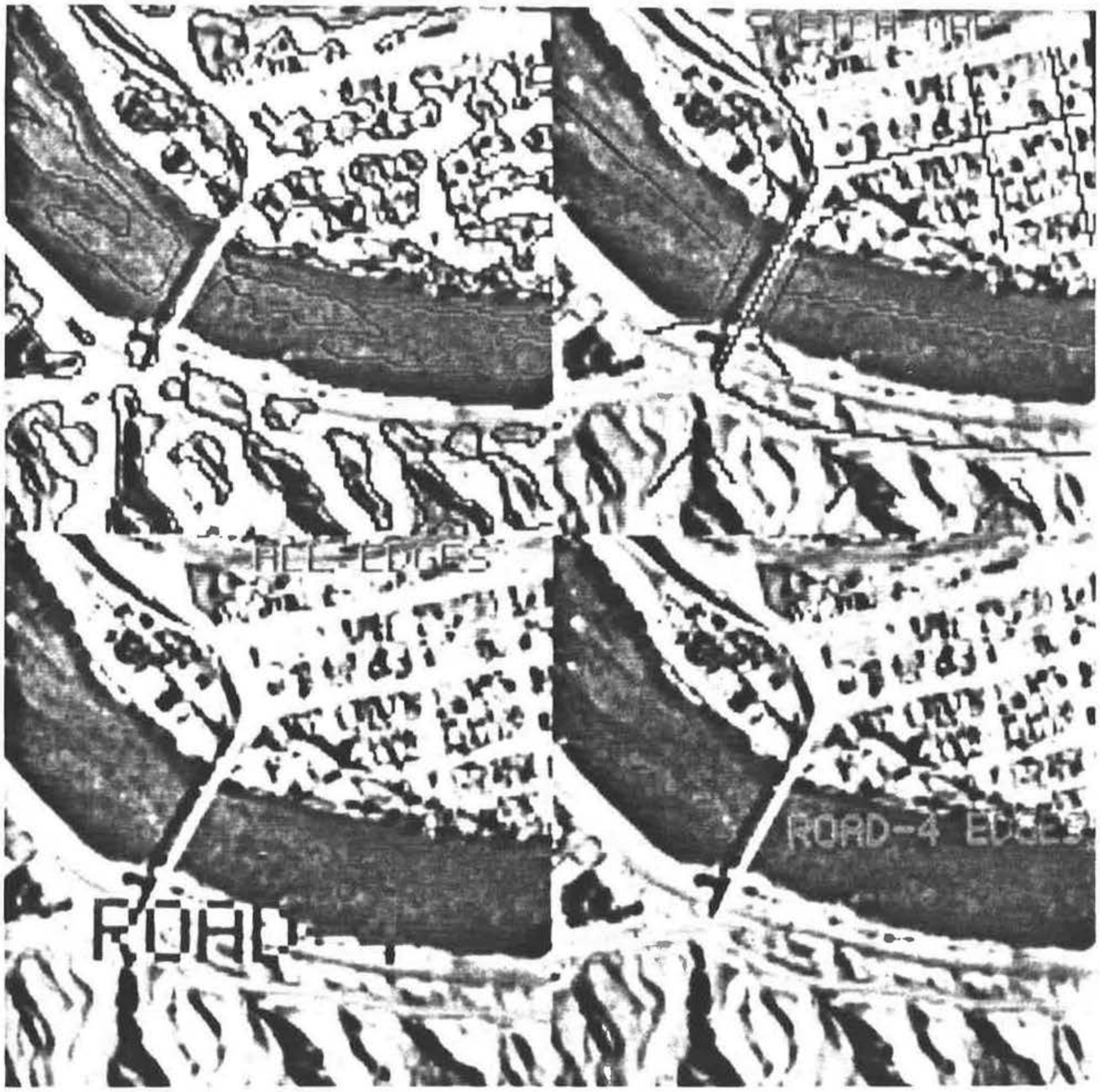
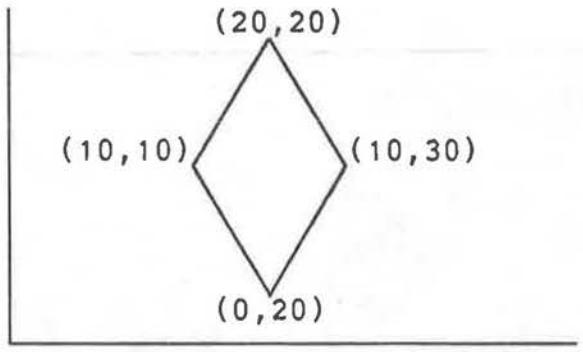


Figure 5.21 Ashcroft: %road-4 Edges



| dist \ σ | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 4 | 2 | 2 | 2 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 8 | 8 | 8 | 8 |
| 5 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 6 | 6 | 6 | 6 | 10 | 10 | 10 |
| 6 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 4 | 8 | 8 | 8 | 8 | 12 | 12 |
| 7 | 2 | 2 | 2 | $\boxed{2}$ | 2 | 2 | 2 | 4 | 6 | 6 | 10 | 10 | 12 | 14 |
| 8 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 4 | 8 | 8 | 12 | 12 | 16 |
| 9 | 2 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 5 | 10 | 14 | 14 | 18 |

σ = 12; distance = 7; number of clusters = 2

| cluster | rightend | numsamples | mean | variance |
|---|---|---|---|---|
| 1 | 91 | 20 | 45.0 | 386.8 |
| 2 | 0 | 20 | 136.0 | 386.8 |

```
0|------------0+++++++++++++|-------------+++++++++++++ 1st.
+|+++++++---------++++++++++|++++++++--------++++++++++ 2nd.
 |            *             |            *
 |            *             |            *
 |            *             |            *
 |            *             |            *
 |            *             |            *
 |            *             |            *
 |*    *   *  * *   *    *  | *   *   *  *  *  *    *
 |*    *   *  * *   *    *  | *   *   *  *  *  *    *
-+--------------------------+--------------------------
0| 1  2  3  4  5  6  7  8  9| 1  1  1  1  1  1  1  1  1
 | 0  0  0  0  0  0  0  0  0| 0  1  2  3  4  5  6  7  8
 |                          | 0  0  0  0  0  0  0  0  0

                               orientation
```

Figure 5.22 Clustering Orientations along the Boundary of a Rectangle

This technique has been used in MISSEE as a predicate to determine if a region in the image is roughly rectangular. The longest stable section in the chart with two clusters is found and the difference of the cluster means is calculated. If it is in the range of (80,110) degrees then the predicate returns true. If no stable section can be found or the difference in means is not close enough to 90 then false is returned.

This predicate is used by the system in two ways. Regions that are instantiated as "city blocks" in towns from the image alone must have the property of rectangularity or they are rejected. And, if the region corresponding to a river is rectangular, then the confidence of that river is raised slightly.

While clustering is used as an inference mechanism in several places within MISSEE, there are other possibilities that could be fruitfully explored. In particular, attempts have not been made to infer global values such as scale from the results of image interpretation, as discussed in Chapter 4.

## 5.6. Control--The Priority Queue, Messages, and Demons

Control is exercised through the cycle of perception (cf. Section 4.4) by means of a global priority queue. Each entry on the queue designates a schema, how it is to be entered, and a possible context for the evaluation of one of its attached procedures. This is analogous to, but more limited than, pattern-directed invocation in Maya and achieves a similar modularity.

Messages are sent from schemata to redirect attention to other schemata as the results of interpretation become available. The user can also initiate messages either to impart new information or to make known his/her requirements. A message is made up of four parts: 1) the name of the generic schema it is directed to, 2) how the schema is being entered (top-down or bottom-up), 3) a priority number, and 4) parameters that are used to establish the context when the appropriate procedure attached to the schema is evaluated. Consequently, the sender of the message (a schema or the user) must "know" two aspects of the receiver. The name of the recipient schema must be known as well as where it sits in the hierarchy relative to the sender. Lateral messages are always sent top-down. A priority value is included to rank the message. The user can provide a value to reflect the importance of his/her message relative to those already in the queue. A schema will generally use its CONFIDENCE value plus or minus some small number to reflect the importance of the message. Finally, some information useful to the procedure, such as location, may be provided.

A function, addtoqueue, adds messages to the queue; see Section 5.4.1.1 for an example. Sometimes it is necessary to modify parts of messages that are on the queue. For example, Section 5.4.2 described how schemata in the hierarchies were sometimes merged when a newly instantiated schema revealed that they were spatially contiguous. If there were any messages whose parameter list included the name of the instance whose identity was submerged, then the new name would replace it on the queue as well. This situation will often arise when the hierarchies are initially being formed.

Another global facility is the demon list. Demons, functions established by attached procedures, are activated when some future condition becomes true. After every attached procedure is executed, each demon is evaluated if its initiation conditions are true. Normally, after a demon has run it will remove itself from the list.

Demons are used in MISSEE in conjunction with lateral messages. While instantiating a bridge, a message may be sent to suggest a region that might correspond to a river. At the same time, a demon will be set up to wait for the river to actually be instantiated. If it is, the demon will create a neighbour link between the two instances and vanish. The demon's action saves the river schema from having to search all of its neighbouring regions for one corresponding to a bridge instance. Section 5.4.1.1 includes an example of a demon being initiated; Appendix E contains an example of one being executed.

The execution cycle is shown in Figure 5.23. Note how interaction with the user can be easily controlled and varied. The global variable %terse is used as the major switch to determine whether interaction should take place. In terse mode, the user is not queried for any initial information. Rather, the default message (td geosystem 100 nil) is sent. Additionally,

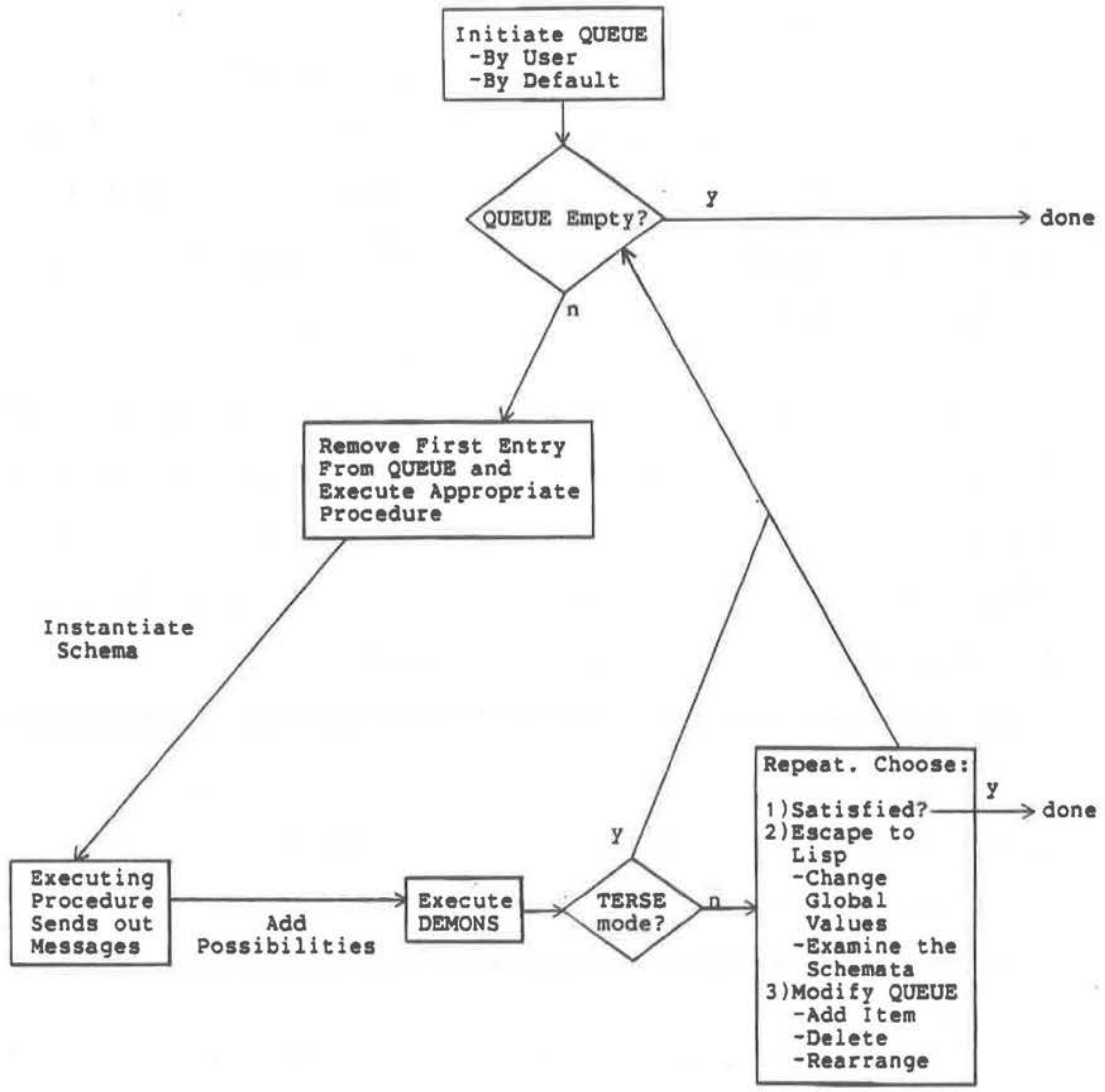


Figure 5.23 The Execution Cycle

the user is not prompted for information as interpretation proceeds. On the other hand, if terse mode is off, then during each cycle the user has a chance to query the system for detailed information (over and above what is provided on the display and in the protocol) and to influence processing in various ways[5]. A new control loop is entered that permits the user to escape to Lisp (to change global values such as the %printlevel, examine schemata, interact with the environment, etc.) or modify the priority queue. When he/she is finished, perhaps without doing anything, then either execution is resumed or the cycle is broken. Examples of interaction within the execution cycle can be found in Appendix E.

This execution cycle appears to be quite different from the cycle of perception (Figure 4.2). However, it controls interpretation in the same manner and each aspect of the cycle of perception can be found somewhere in the execution loop. Both provide for the timely instantiation of relevant schemata and maintain a focus of attention.

[5]One might wonder how the user can get out of terse mode when no interaction is taking place. In a Lisp environment, it is possible to break the execution, execute a command such as (setq %terse nil), and then continue the execution where it left off.

# CHAPTER 6

# Results and Evaluation

MISSEE has been tested with 6 images of small urban areas. The results indicate the usefulness of the ideas previously described. This chapter summarizes the results and indicates how they meet the criteria of success described in Chapter 3.

The Ashcroft image and some results of its interpretation have been used in previous chapters for illustrative purposes. Figures 6.1, 6.2, and 6.3 show the other three images that have been used: Houston, Spences Bridge, and Cranbrook, all of which are in British Columbia. Two subimages of the Spences Bridge photograph have been extracted to show that MISSEE is robust with respect to scale differences--four to one in this case. The extracted images will be called Spences Bridge West and Spences Bridge East to distinguish them. The information sources for all the images can be found in Figures 6.4 through 6.9.

## 6.1. Instantiating Objects With the Aid of a Sketch Map

Except in the case where the user is controlling every step of the instantiation process, the best results should appear when a sketch map is used in addition to the intensity image. Of course, only those objects represented in the sketch map will receive guidance, but since the user probably included all of

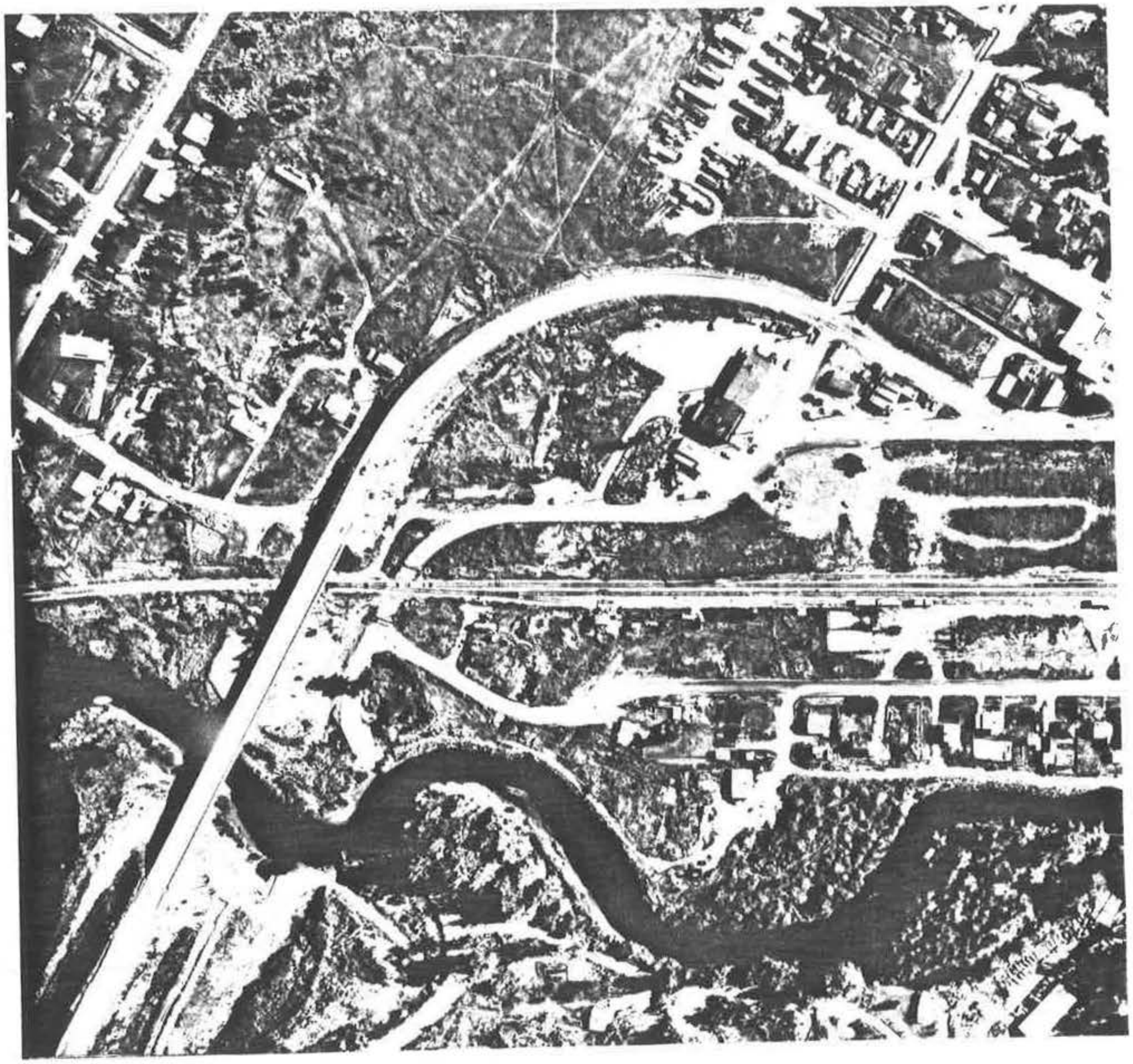


Figure 6.1 Houston, British Columbia

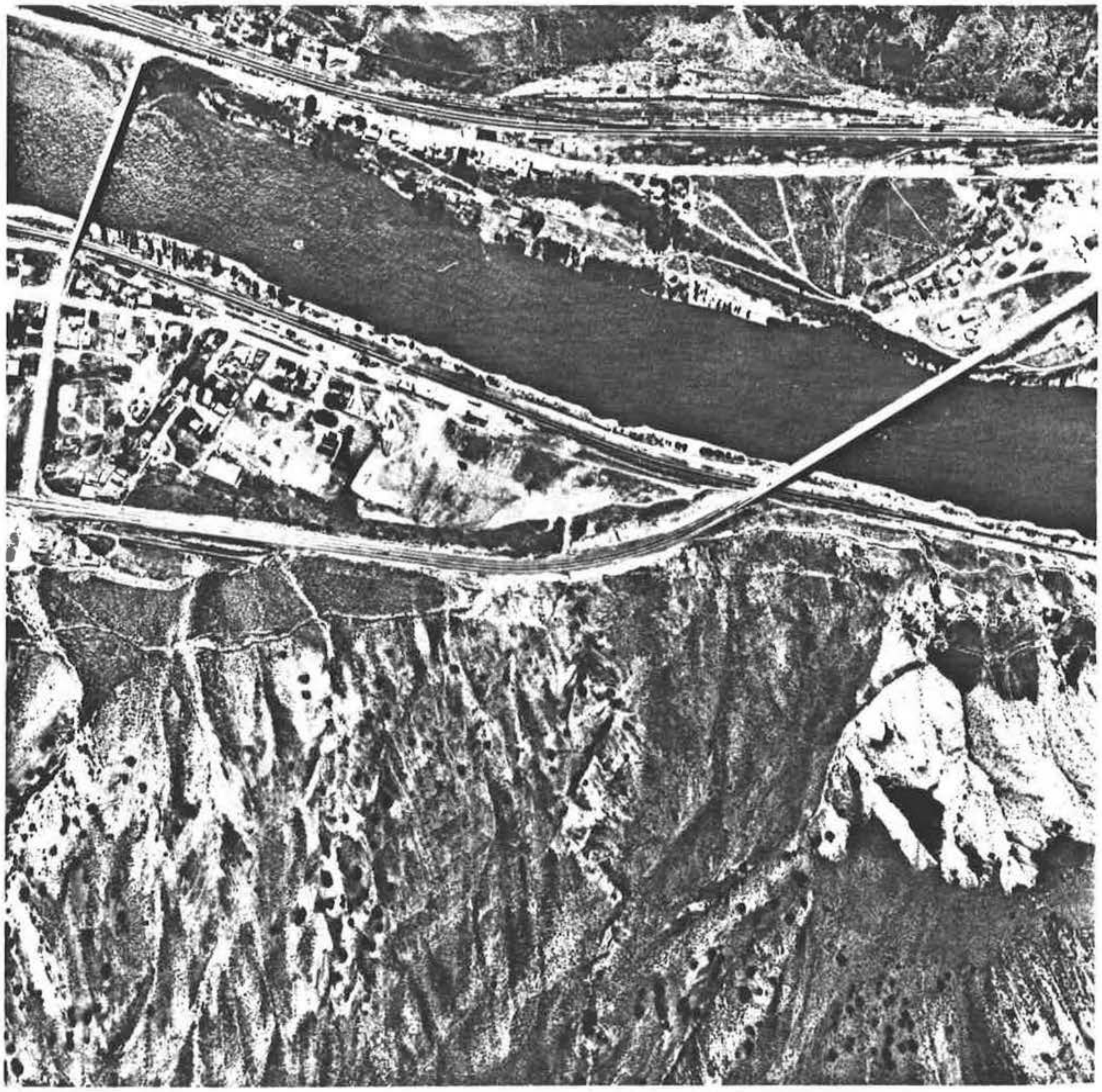


Figure 6.2 Spences Bridge, British Columbia

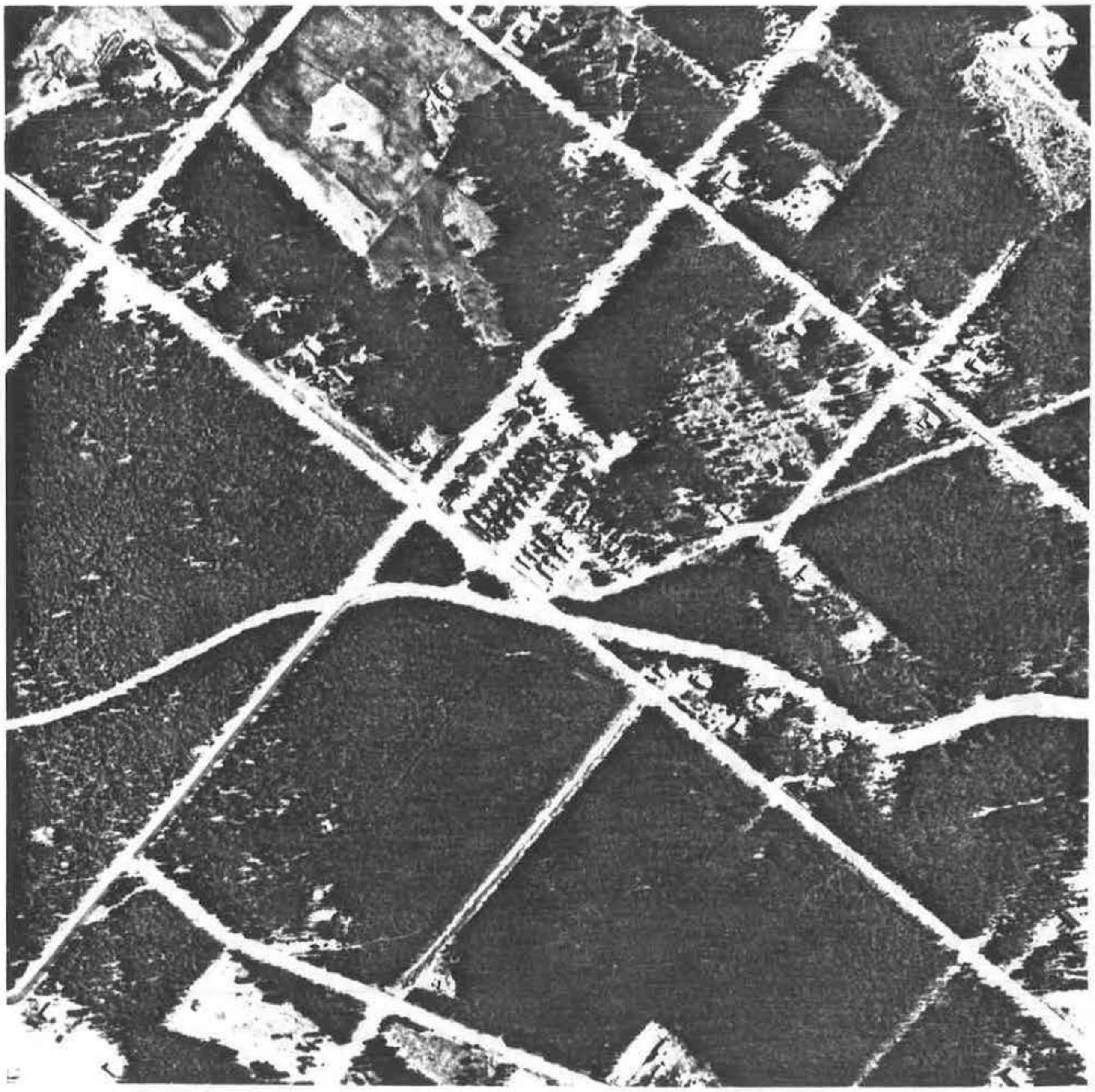


Figure 6.3 Cranbrook, British Columbia

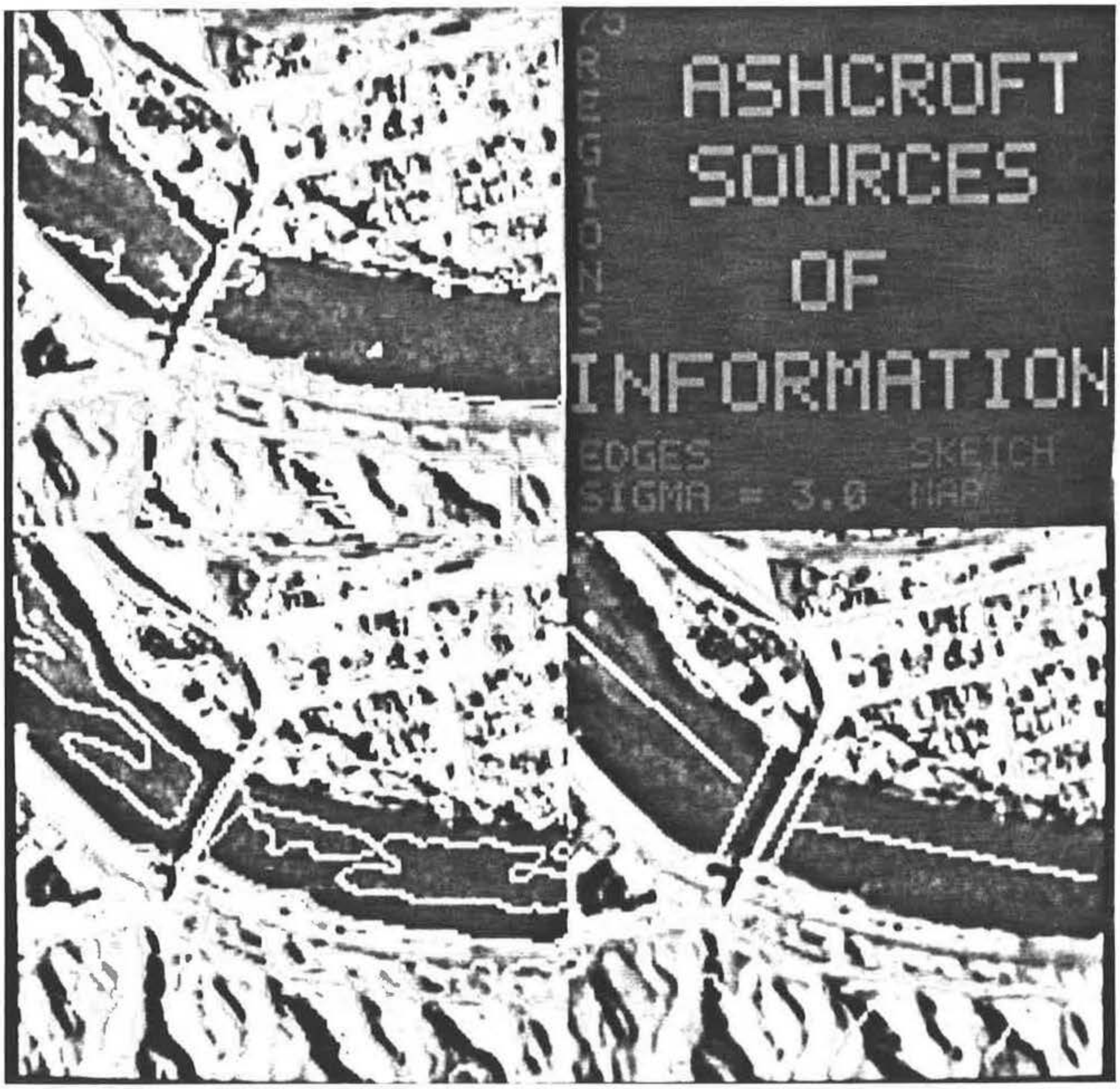


Figure 6.4 Ashcroft: Information Sources

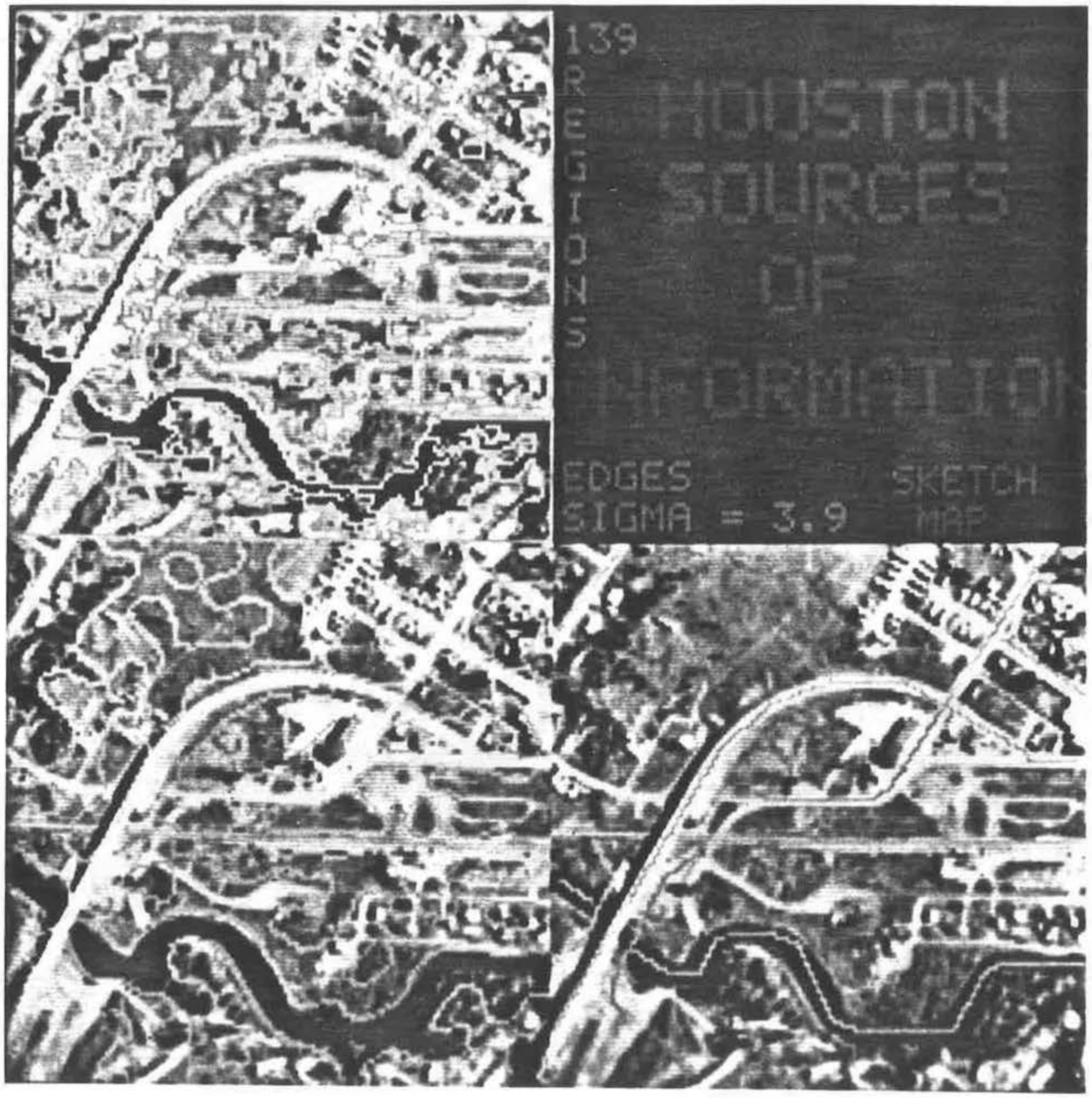


Figure 6.5 Houston: Information Sources

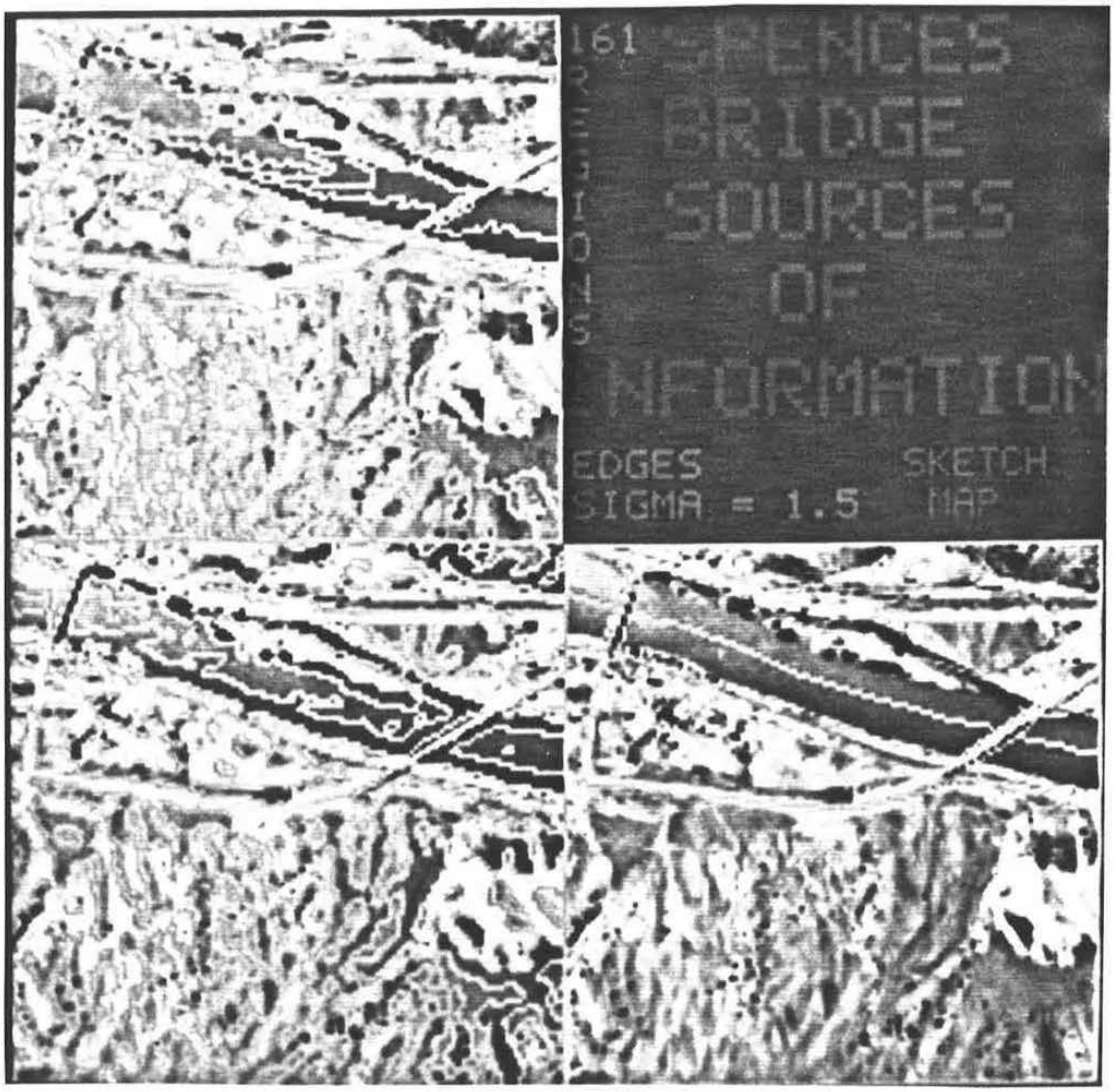


Figure 6.6 Spences Bridge: Information Sources

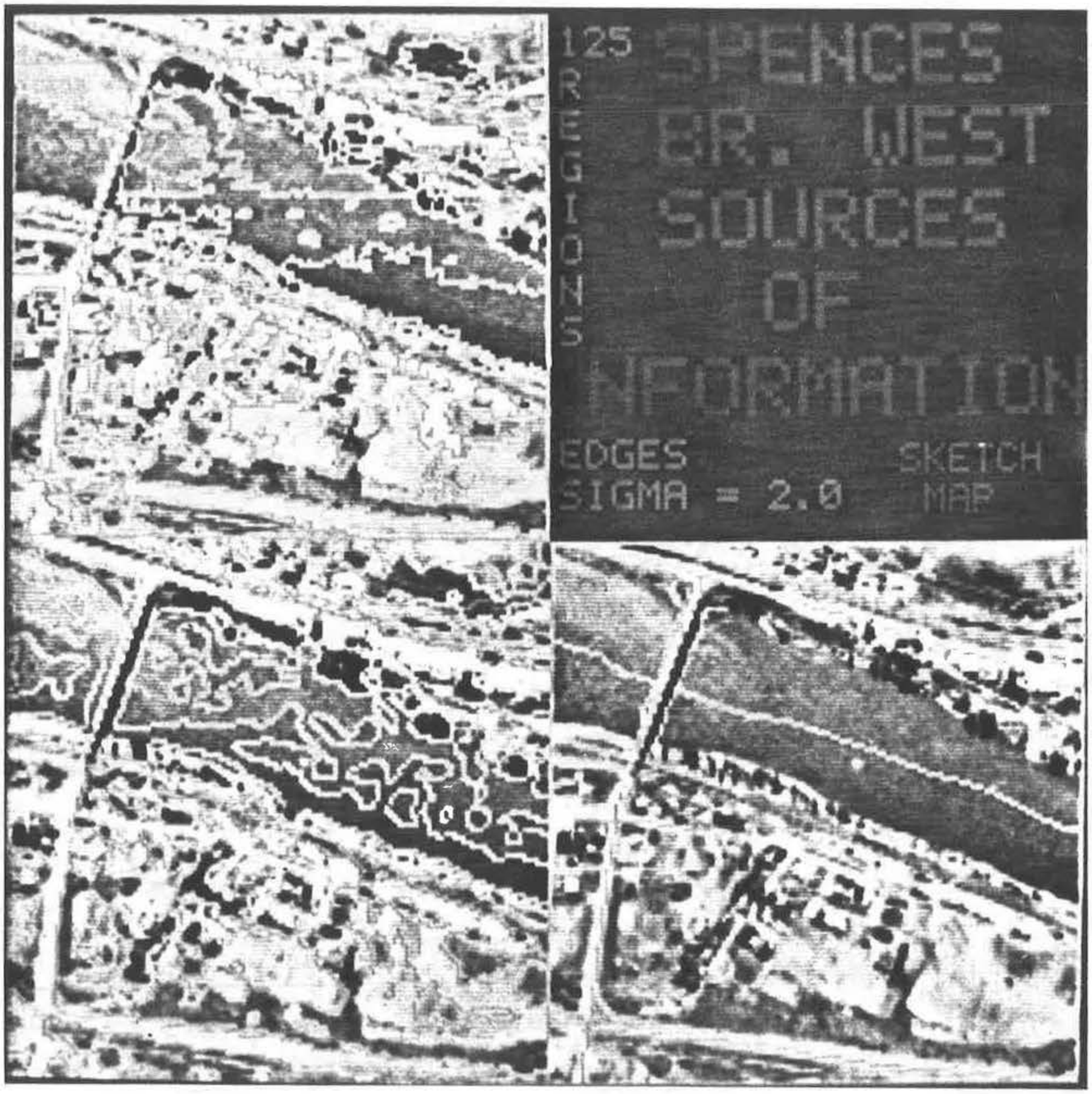


Figure 6.7 Spences Bridge West: Information Sources

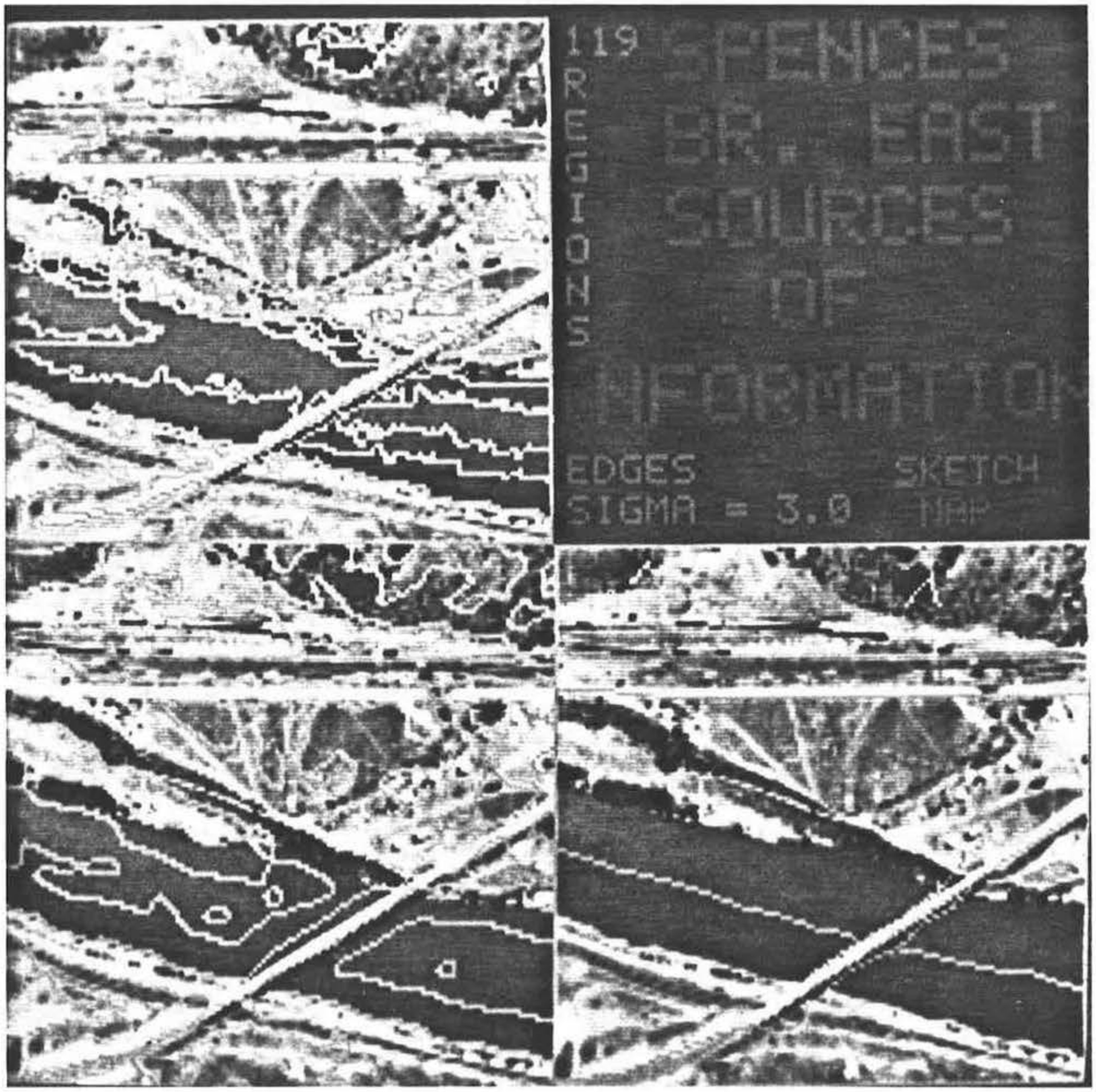


Figure 6.8 Spences Bridge East: Information Sources

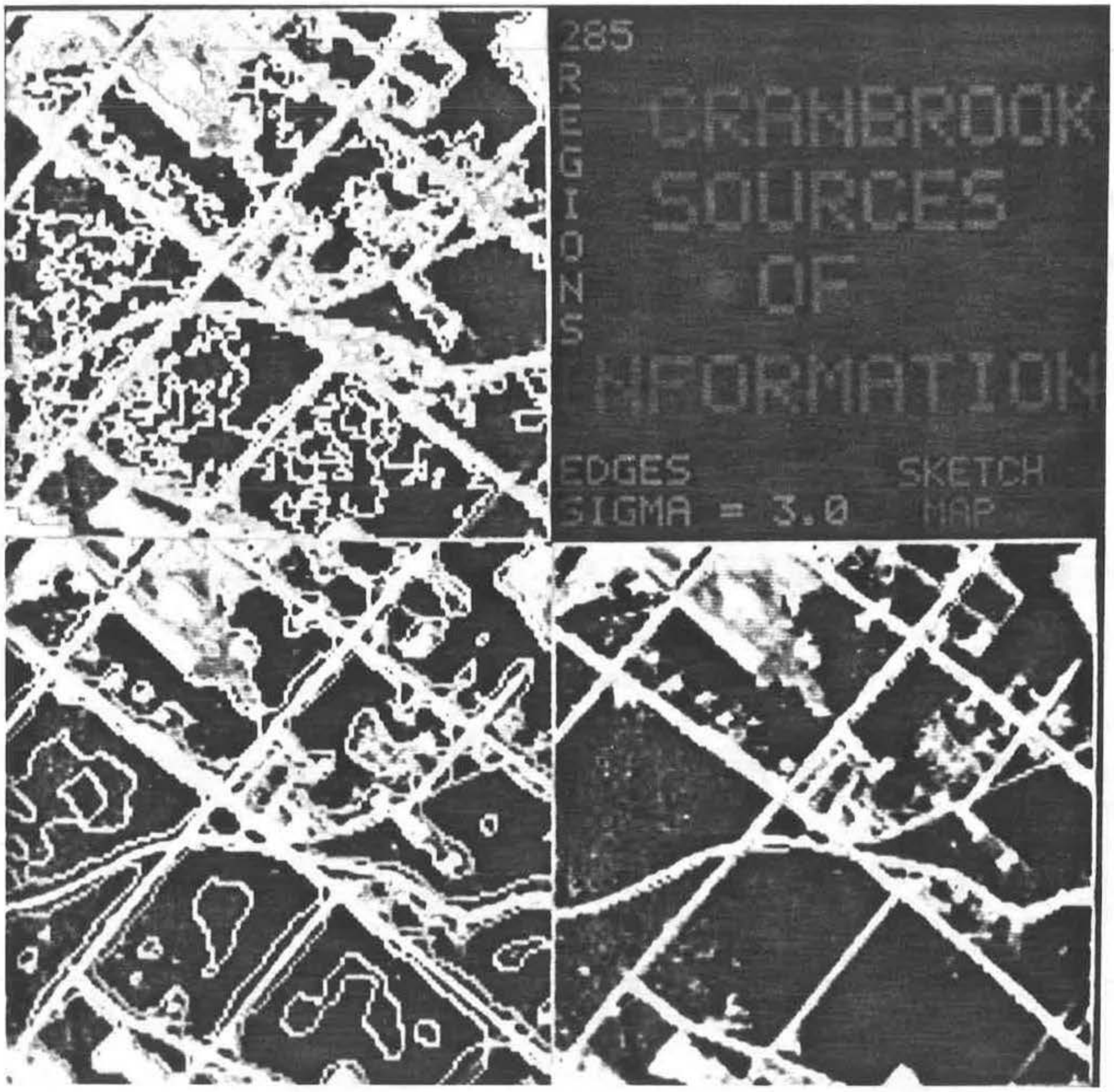


Figure 6.9 Cranbrook: Information Sources

the items of interest, that should not be a limitation.

Two types of errors can result when the procedure associated with a schema that knows how to take advantage of sketch map instances tries to find a corresponding element in the digitized image. Recall that chains in the sketch map may be ambiguously interpreted so that there is one "intended" interpretation (i.e. as the person who drew it intended) and (possibly) several that are "unintended". Thus two erroneous situations can occur: 1) the intended model could not be instantiated in the image, or 2) an unintended one could. In the terms used in Section 3.2, these would generally be cases of unsatisfiability and ambiguity, respectively.

The possible interpretations of the Ashcroft sketch map were shown in Table 5.1. Table 6.1 divides the instances into those that represent the interpretation intended by the person who drew the sketch map and those that were produced because of ambiguities in the shapes of the chains. The table also reveals which of the sketch map instances were able to guide the instantiation of their counterparts in the image. They are printed in upper case while instances that were unable to find support in the image are in lower case. Thus, ideally, all the intended interpretations in the left column would be in upper case letters and unintended interpretations would be printed in lower case.

Two errors of the second type come from chains 12 and 13 whose road schemata were able to find agreement in the image. This is not surprising since roads and mountains both search for bright regions and straight edges (ridges or curbs). If the ridges happen to pair off at a certain distance, then they will be mistaken for roads. However, chains that correspond to roads will very rarely have the sketch map interpretation of "mountain", so that a heuristic can be used to remove those "unintended" road interpretations.

Table 6.2 summarizes the situation for Houston. Here there is an error of the first type. *road-5 has failed to find the proper features to be instantiated in the image as a road[1]. This situation arises because the region merger has joined the

| CHAIN | INTENDED INTERPRETATION | UNINTENDED INTERPRETATION(S) |
|---|---|---|
| 3,5 | *BRIDGE-1 | |
| 4 | *TOWN-1 | |
| 9 | *RIVER-1 | *road-1 |
| 11 | *RIVER-2 | *road-2 |
| 10 | *ROAD-3 | *river-3 |
| 2 | *ROAD-4 | *river-4 |
| 6 | *ROAD-5 | *river-5 |
| 1 | *ROAD-6 | *river-6 |
| 7 | *ROAD-7 | *river-7 |
| 8 | *ROAD-8 | *river-8 |
| 12 | *MOUNTAIN-1 | *ROAD-9, *river-9 |
| 13 | *MOUNTAIN-2 | *ROAD-10, *river-10 |
| 14 | *MOUNTAIN-3 | *road-11, *river-11 |

Table 6.1 Ashcroft: Sketch Map to Image

[1]Chain-5 is the one in the very top right of the sketch map.

region corresponding to the road with the one next to it and the resulting average intensity has dropped below the threshold for acceptance.

Similar results are shown for Spences Bridge, Spences Bridge West, Spences Bridge East, and Cranbrook in Tables 6.3 through 6.6. In summary, out of 100 sketch map instances there were 3 errors of the first type and 6 errors in the second

| CHAIN | INTENDED INTERPRETATION | UNINTENDED INTERPRETATION(S) |
|---|---|---|
| 2,4 | *BRIDGE-1 | |
| 8 | *RIVER-1 | *road-1 |
| 9 | *RIVER-2 | *road-2 |
| 1 | *ROAD-3 | *river-3 |
| 6 | *ROAD-4 | *river-4 |
| 5 | *road-5 | *river-5 |
| 3 | *ROAD-6 | *river-6 |
| 7 | *ROAD-7 | *RIVER-7, *bridge-2 |

Table 6.2 Houston: Sketch Map to Image

| CHAIN | INTENDED INTERPRETATION | UNINTENDED INTERPRETATION(S) |
|---|---|---|
| 13 | *town-1 | |
| 12 | *ROAD-1 | |
| 9 | *ROAD-2 | |
| 11 | *ROAD-3 | |
| 10 | *ROAD-4 | |
| 5,6 | *BRIDGE-1 | |
| 2 | *RIVER-5 | |
| 3 | *RIVER-6 | |
| 7,8 | *BRIDGE-2 | |
| 4 | *RIVER-7 | |
| 14 | *MOUNTAIN-1 | *ROAD-8, *RIVER-8 |
| 15 | *MOUNTAIN-2 | *road-9, *river-9 |
| 16 | *MOUNTAIN-3 | *road-10, *RIVER-10 |

Table 6.3 Spences Bridge: Sketch Map to Image

| CHAIN | INTENDED INTERPRETATION | UNINTENDED INTERPRETATION(S) |
|---|---|---|
| 4,5 | *BRIDGE-1 | |
| 2 | *RIVER-1 | |
| 3 | *RIVER-2 | |
| 10 | *TOWN-1 | |
| 9 | *ROAD-3 | |
| 6 | *ROAD-4 | |
| 7 | *ROAD-5 | *river-5 |
| 8 | *ROAD-6 | *river-6 |

Table 6.4 Spences Bridge West: Sketch Map to Image

| CHAIN | INTENDED INTERPRETATION | UNINTENDED INTERPRETATION(S) |
|---|---|---|
| 4,5 | *BRIDGE-1 | |
| 2 | *RIVER-1 | |
| 3 | *RIVER-2 | |
| 6 | *ROAD-3 | |
| 7 | *ROAD-4 | *river-4 |
| 8 | *MOUNTAIN-1 | *river-5, *road-5 |
| 9 | *mountain-2 | *river-6, *road-6 |

Table 6.5 Spences Bridge East: Sketch Map to Image

| CHAIN | INTENDED INTERPRETATION | UNINTENDED INTERPRETATION(S) |
|---|---|---|
| 5 | *ROAD-1 | *river-1 |
| 4 | *ROAD-2 | *river-2 |
| 8 | *ROAD-3 | *river-3 |
| 6 | *ROAD-4 | *river-4 |
| 9 | *ROAD-5 | *river-5 |
| 3 | *ROAD-6 | *river-6 |
| 2 | *ROAD-7 | *river-7 |
| 7 | *ROAD-8 | *river-8 |

Table 6.6 Cranbrook: Sketch Map to Image

category. Overall, this is a 9% error rate.

Visual evidence also indicates that appropriate image features were matched with MISSEE instances. Figures 5.15 and 5.16 displayed the regions and edges that were found for the bridge in the Ashcroft image. The following figures show similar results for the other "bottom level" objects in the Ashcroft image. Regions are the prominent features for towns (Figure 6.10) and rivers (6.11). Mountains are associated with edges (6.12). Roads, like bridges, map to both edges (6.13 and 6.14) and regions (6.15). The region for roads 4, 5, and 6 is the same as for roads 1 and 2.

## 6.2. Instantiating Objects from the Intensity Image Alone

Geographic objects can be instantiated with some success from just the digitized image. However, the situation is more ambiguous without the aid of a sketch map and enough evidence is often discovered to instantiate many entities with low confidence.

Roads and bridges are computationally much harder to find using images alone because the regions that correspond to them have generally merged together many individual roads. Having found a suitable region, the program searches for bounding water regions for bridges (an example is in Section 5.4.1.2) or suitably spaced parallel edges for roads. With roads, this tends to find city blocks, from one intersection to another, rather than continuous streets. Thus, besides being less efficient, the results may be different even when they are correct.

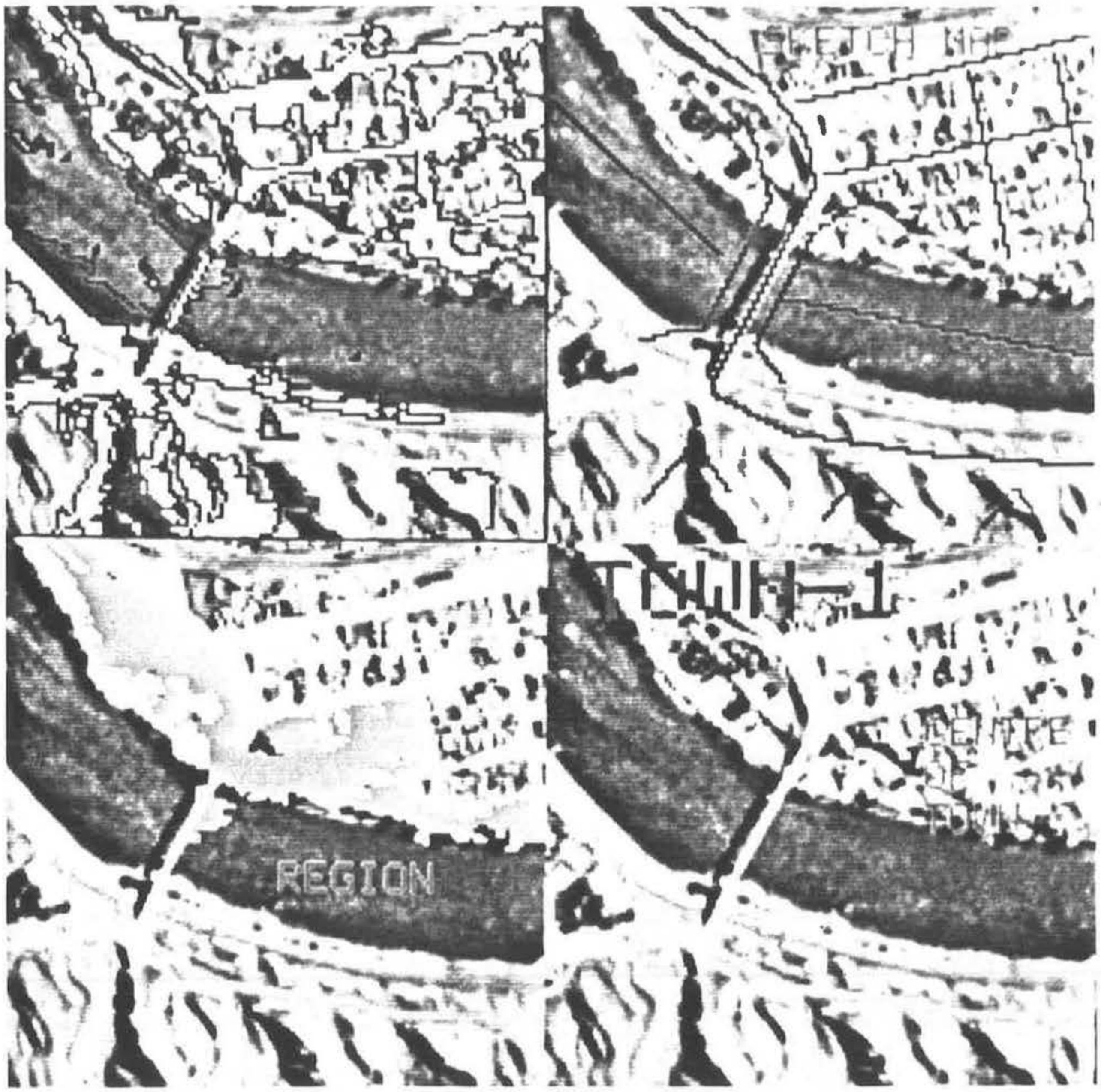


Figure 6.10 Ashcroft: Sketch Map. %town-1

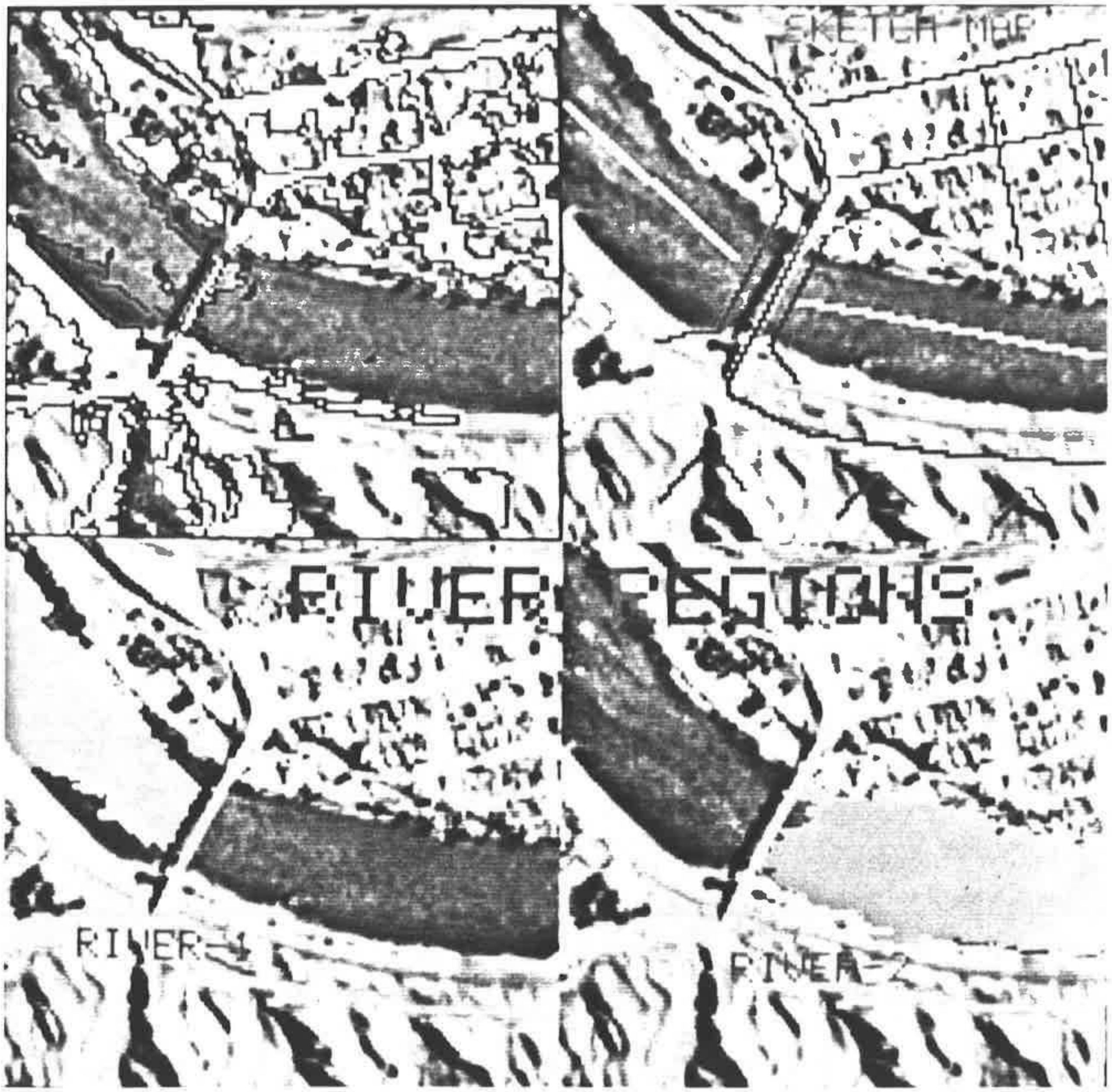


Figure 6.11 Ashcroft: Sketch Map. River Regions

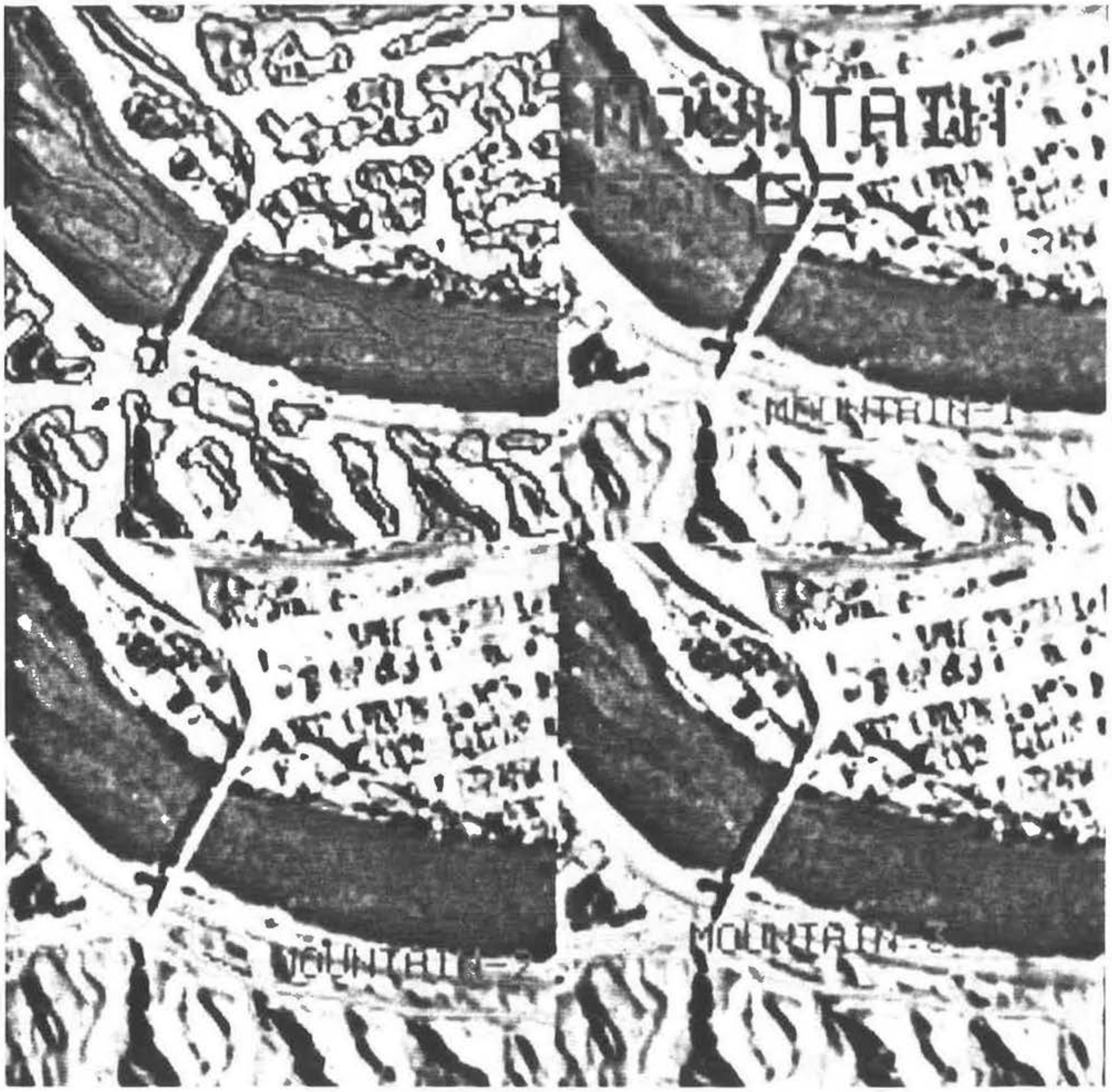


Figure 6.12 Ashcroft: Sketch Map. Mountain Edges

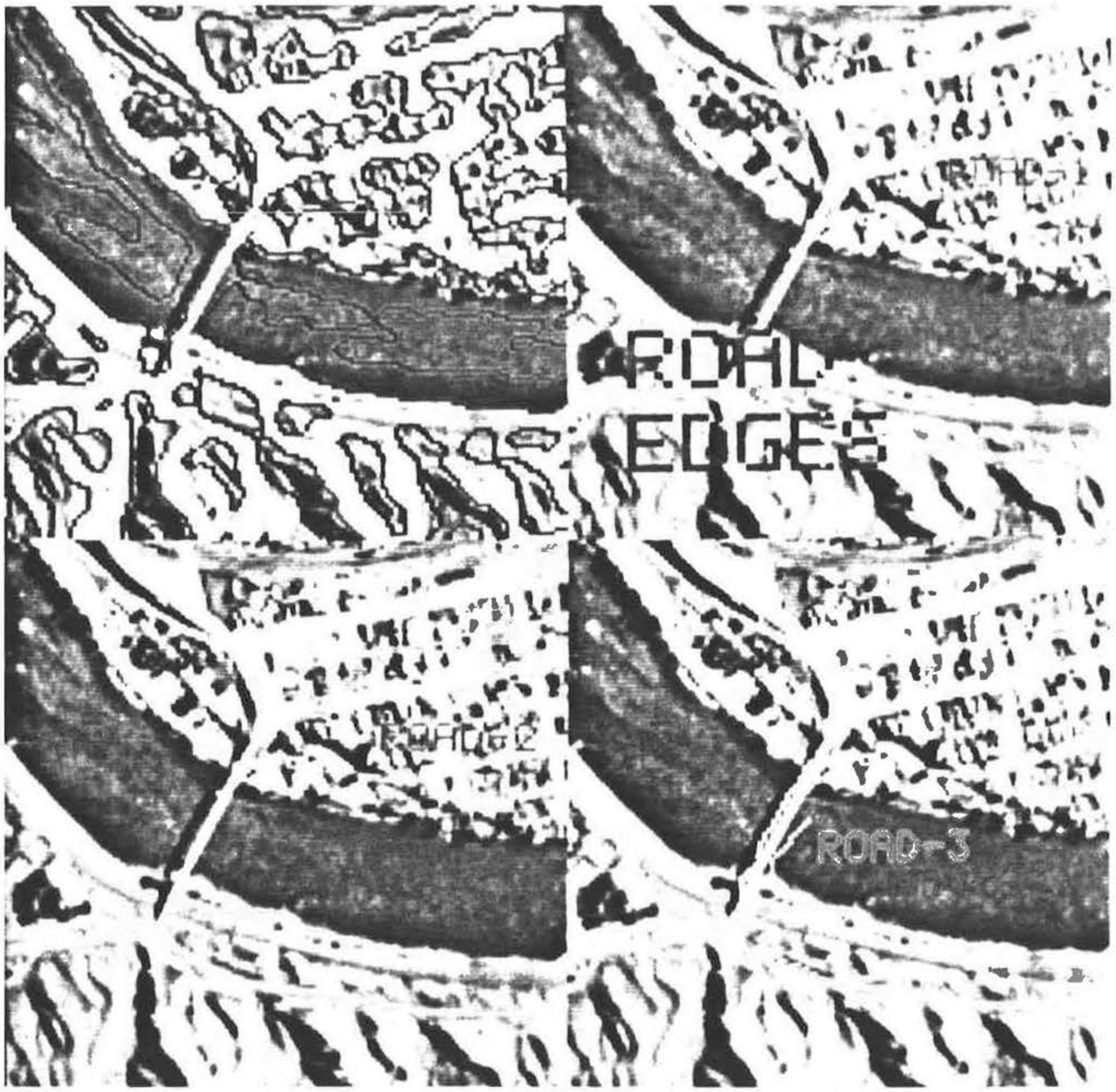


Figure 6.13 Ashcroft: Sketch Map. Roads 1-3, Edges

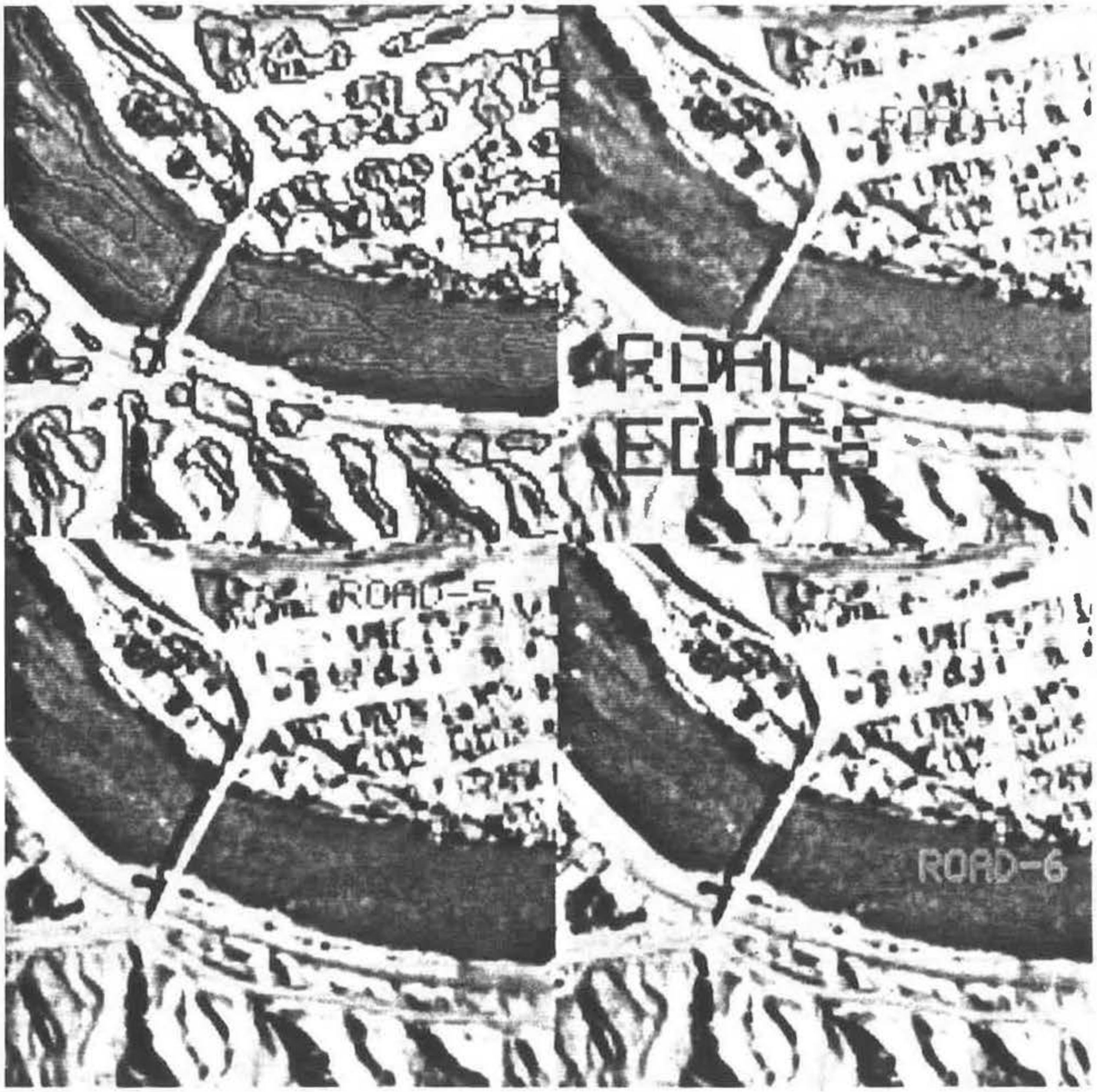


Figure 6.14 Ashcroft: Sketch Map. Roads 4-6, Edges

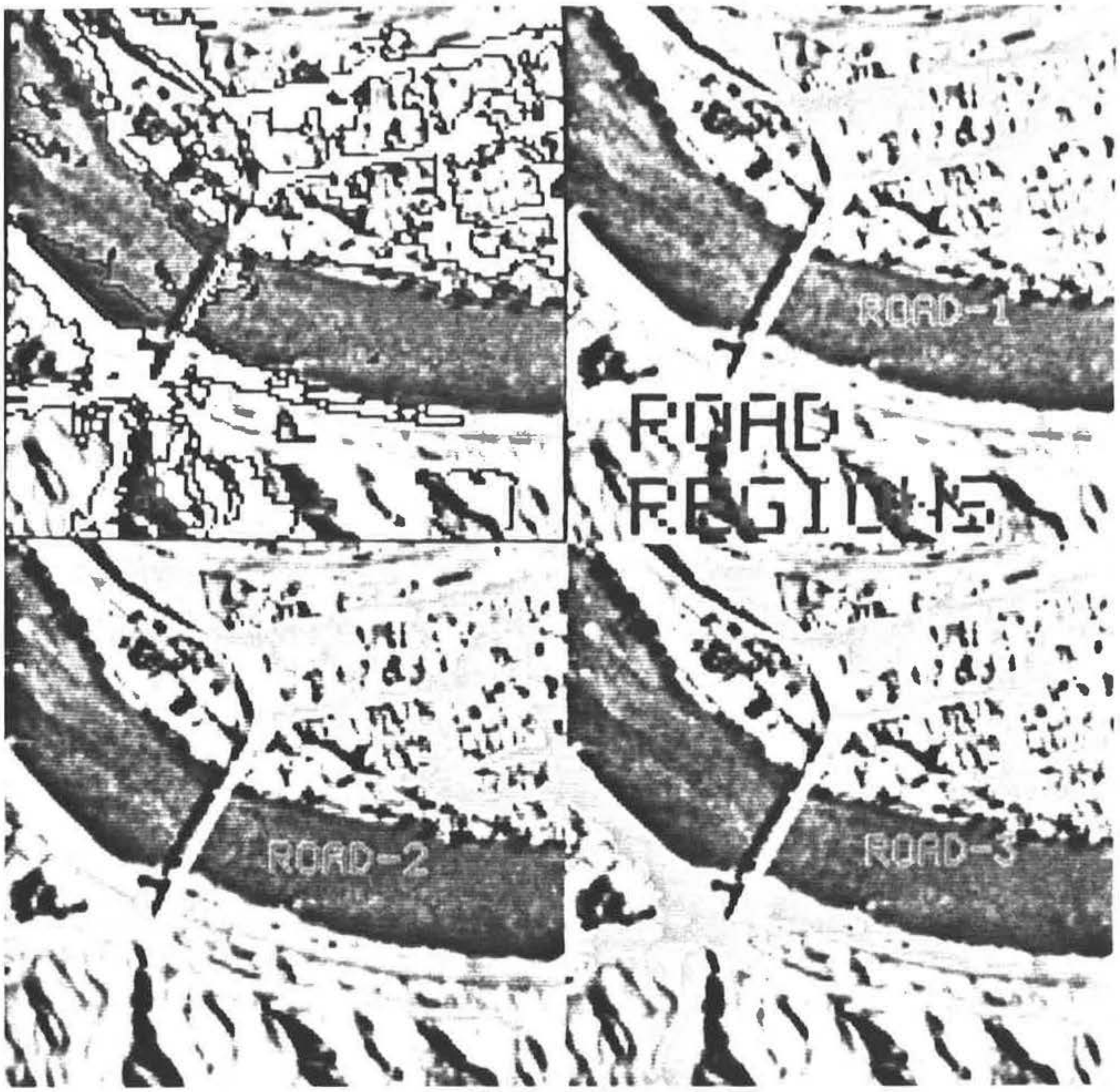


Figure 6.15 Ashcroft: Sketch Map. Roads 1-3, Regions

There is a slight mismatch in the semantics of "town" as found in sketch maps versus the images analyzed by MISSEE. To Mapsee, a town is just that--the whole town--and roads are all interurban thoroughfares. In the images that are being examined here, one is looking inside the town at roads, rivers, etc. Thus the sketch map "blob" has been used to represent "downtown", or the core of the city. That is a rather arbitrary denotation, however, and when considering images alone, town refers to areas that are urban (near roads, containing buildings). Thus one would expect to see only one or two towns in the sketch map, but several in the image.

The next few figures compare the results of an interpretation using the image alone and one with the aid of a sketch map. %road-2 (Figure 6.16) was instantiated with the aid of the Ashcroft sketch map and contains 19 edges. Four roads instantiated from the image alone (Figure 6.17) are required to obtain similar results. However, sometimes the results from the two techniques are similar as is shown in Figures 5.21, where %road-4 is instantiated with the aid of the sketch map, and 6.18, where %road-12 is instantiated from the image alone. The one region corresponding to downtown in the Ashcroft sketch map was shown in Figure 6.10. That region was not acceptable when the image alone was examined because it is not rectangular. This is illustrated in Figure 6.19 which contains several urban regions.

The results for the interpretation of the 5 bottom level objects from the images alone are summarized in Table 6.7.

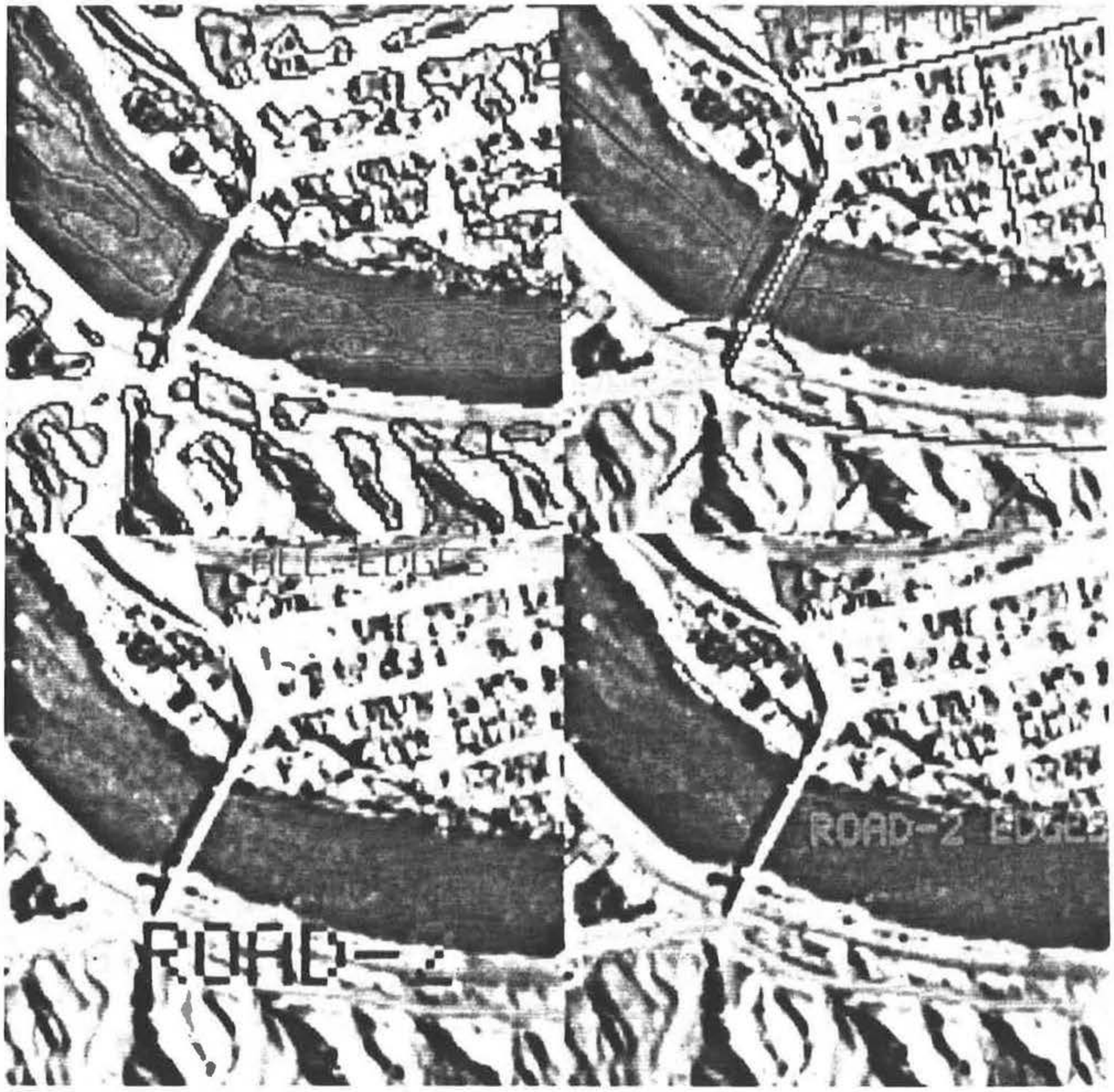


Figure 6.16 Ashcroft: Sketch Map. %road-2

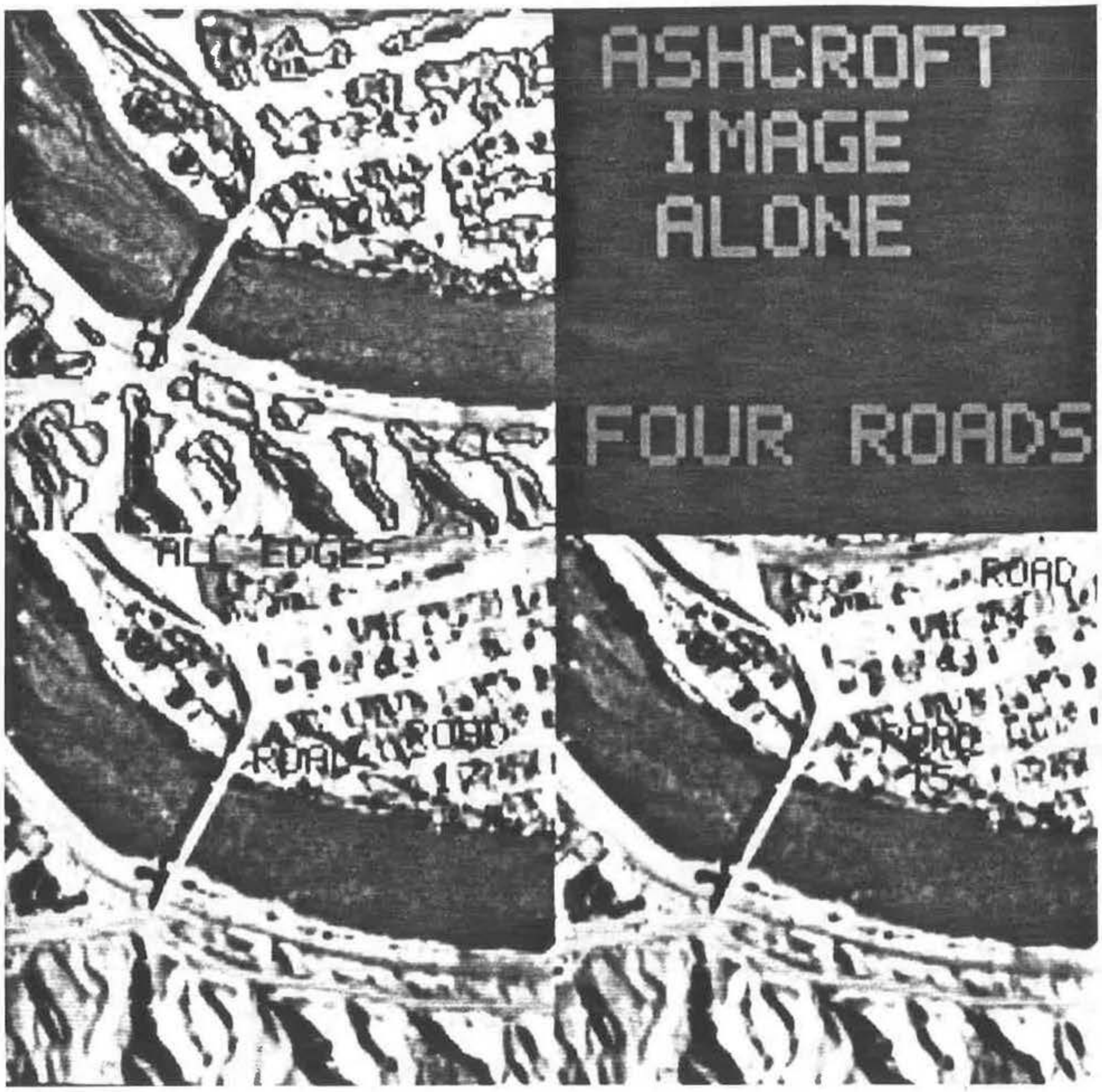


Figure 6.17 Ashcroft: Image Alone. Four Roads

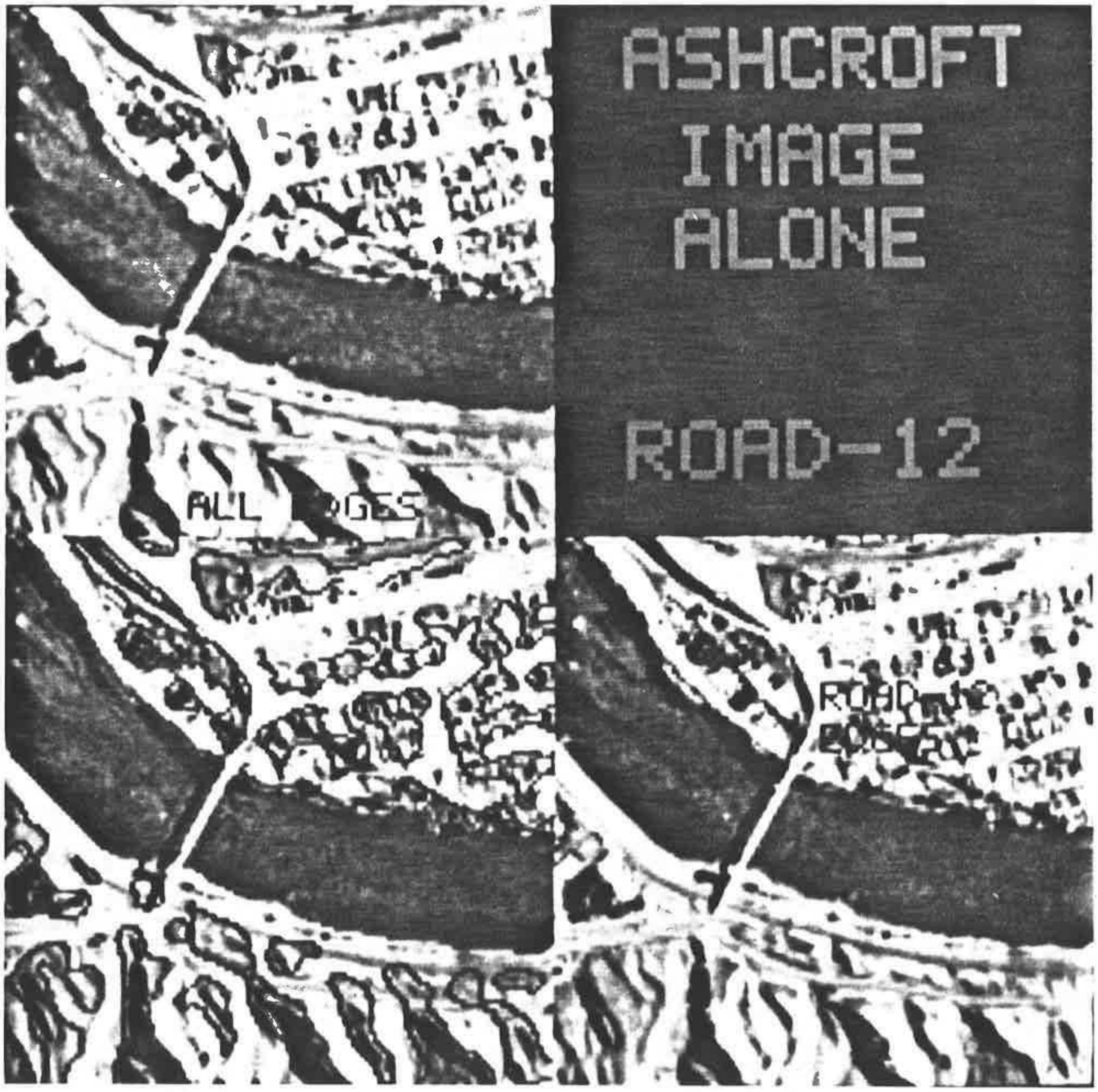


Figure 6.18 Ashcroft: Image Alone. %road-12

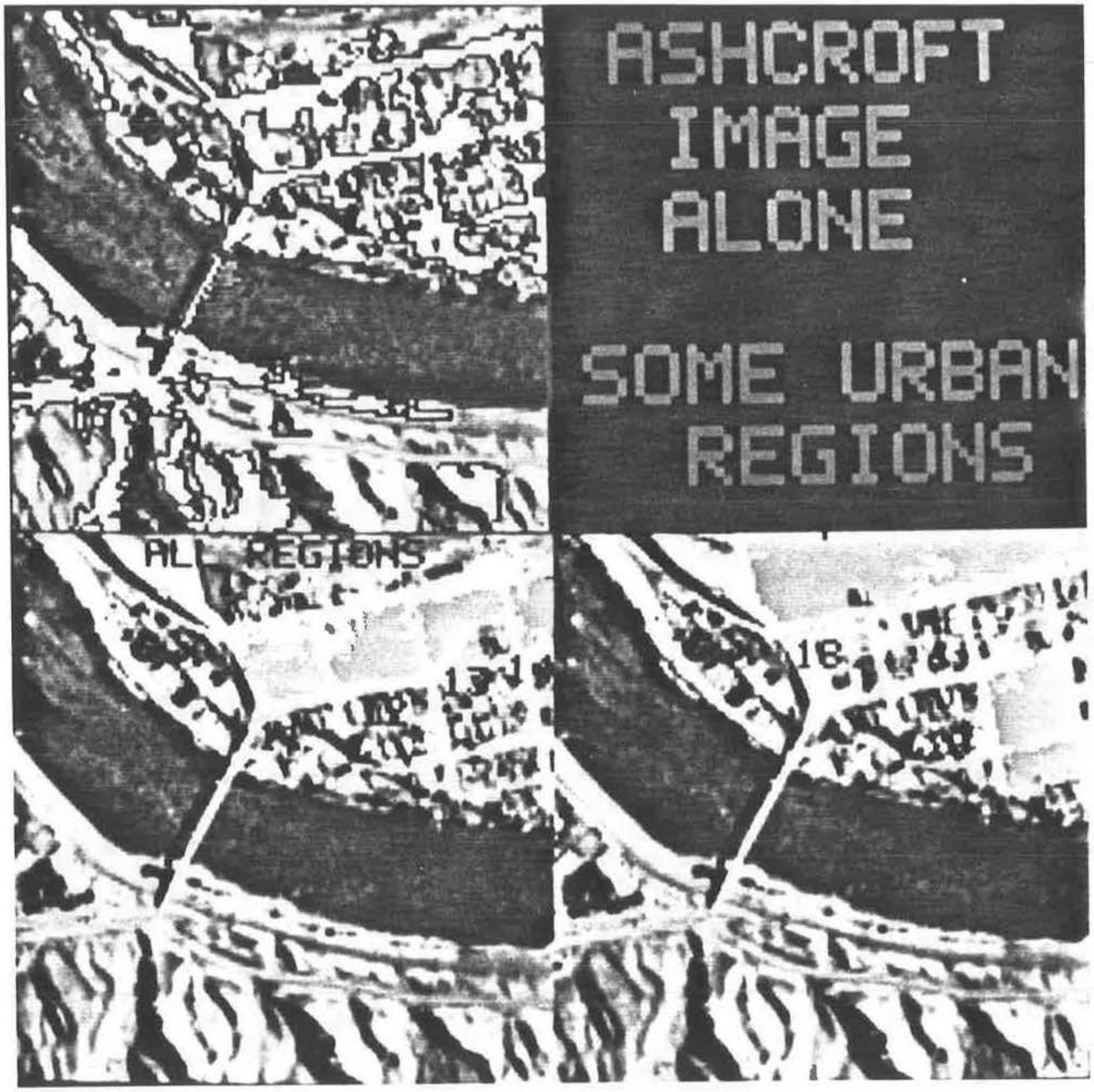


Figure 6.19 Ashcroft: Image Alone. Several Urban Regions

| | BRIDGE | MOUNTAIN | RIVER | ROAD | TOWN |
|---|---|---|---|---|---|
| Ashcroft | | | | | |
| CORRECT | 1 | 3 | 2 | 30 | 7 |
| INCORRECT | 0 | 2 | 0 | 11 | 5 |
| OMITTED | 0 | 2 | 0 | 1 | 1 |
| Houston | | | | | |
| CORRECT | 1 | 0 | 2 | 20 | 26 |
| INCORRECT | 1 | 0 | 0 | 1 | 1 |
| OMITTED | 0 | 0 | 0 | 5 | 5 |
| Spences Bridge | | | | | |
| CORRECT | 2 | 3 | 2 | 51 | 4 |
| INCORRECT | 10 | 0 | 3 | 14 | 0 |
| OMITTED | 0 | 1 | 0 | 0 | 1 |
| Spences Bridge West | | | | | |
| CORRECT | 0 | 4 | 2 | 81 | 2 |
| INCORRECT | 7 | 0 | 2 | 0 | 1 |
| OMITTED | 1 | 0 | 0 | 0 | 0 |
| Spences Bridge East | | | | | |
| CORRECT | 0 | 0 | 2 | 12 | 4 |
| INCORRECT | 1 | 0 | 0 | 2 | 0 |
| OMITTED | 1 | 2 | 0 | 0 | 0 |
| Cranbrook | | | | | |
| CORRECT | 0 | 0 | 0 | 7 | 6 |
| INCORRECT | 0 | 0 | 0 | 0 | 0 |
| OMITTED | 0 | 0 | 0 | 4 | 1 |

Table 6.7 All Images: Instantiation Results

There are again two types of errors: instances that have been instantiated that should not have been (called incorrect); and, objects in the image that did not have the right features to cause instantiation (omitted). The former error is an incorrect match between model and data (as opposed to a numeric mismatch between the domain and the range); the latter error is one of incompleteness.

Overall, the error rate for interpretation from the image alone is (#INCORRECT + #OMITTED) / #TOTAL = 24%. As expected, this is higher that the error rate when a sketch map is also used (9%).

## 6.3. Building Up Hierarchies

When the first bottom level schemata are instantiated, they cause (via messages) the entire hierarchy above them to be created. As more schemata are instantiated (possibly using the results of earlier processing), they either fit directly into the existing network, or cause the creation of a subgraph that can. This entry into the network is mediated by the neighbour relation which determines which scene elements are located next to each other.

The neighbour relationships are established either by demons or by the schema-specific routines after the existence of a new instance has been confirmed. Each object has information about its possible neighbours and ways to establish the link. Generally, it involves finding common edges or regions or neighbouring regions in the appropriate instances. When the link is established, the inverse relation (also "neighbour", since the relation is reflexive) is also created.

Neighbouring instances of the same type will have a common parent node up at least one of the hierarchies. Sometimes a new instance will indicate that it relates two or more groups. In that case, the parent nodes are all collapsed into one of the nodes.

The results of this hierarchy building are based on whether the neighbour relation is established. When it is properly formed, the hierarchies will always be built up correctly. If some neighbour relationships are not found but some of the links are redundant, the hierarchies may still be correctly formed. However, if enough crucial links are missing, then the hierarchies will be split into separate (sub)graphs.

The formation of neighbour links can suffer from two types of errors: 1) an incorrect link can be made between non-neighbouring schemata, or 2) the link might not be found. In MISSEE, the former error has never occurred and the latter only rarely. Furthermore, as was just explained, errors of omission are less problematic because neighbour links may be redundant. Hence, it depends on which schemata were actually instantiated (and the values used to fill their slots), which in turn depends on the user's strategy.

Figure 5.19 showed the relations (except instance of) that would be established if all of the Ashcroft sketch map instances were used to aid in interpreting the image. There are no errors in this graph. Figure 6.20 shows what would happen if only the Ashcroft image was used. Only the correct instantiations are included and several of the intervening features required to

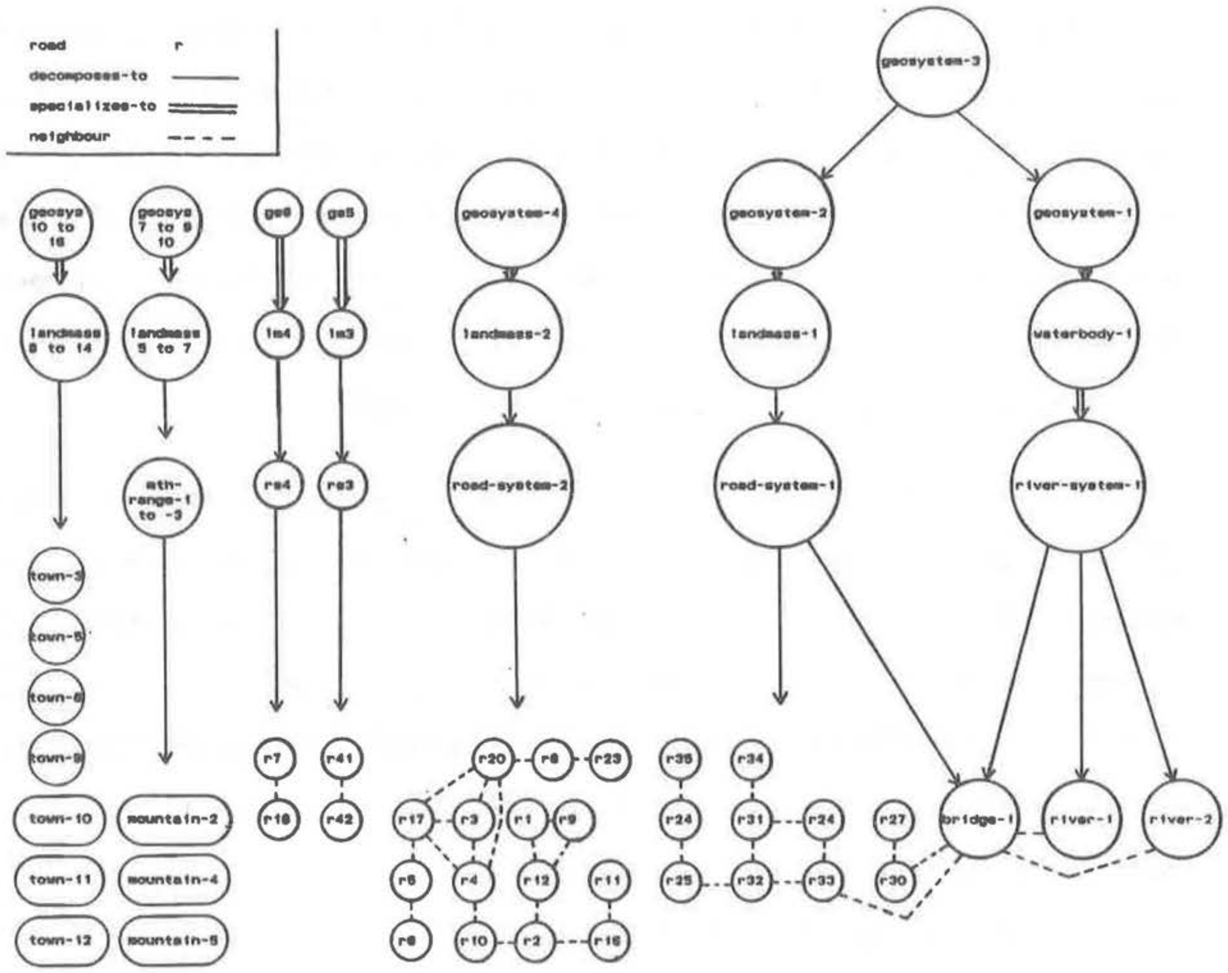


Figure 6.20 Ashcroft: Instance Hierarchy 2

establish neighbour links are missing (especially for roads). Thus the graph is fragmented into 13 pieces more than the one in Figure 5.19.

Appendix E contains a protocol that shows a session with the Ashcroft image where the user interacts with the system. All types of schema-specific routines are employed. The resulting hierarchy is shown in Figure 6.21. There are no errors in this graph which contains fewer instances than Figure 5.19, only because the user terminated processing with possibilities remaining.

Similar results have been obtained for the other images. Figures 6.22 and 6.23 show the relationships for Houston and Spences Bridge that were derived automatically using the sketch map to guide interpretation.

## 6.4. Moving Up and Down the Hierarchies

The goal of the cycle of perception is to use the results of interpretation to guide further processing so that scene knowledge is accumulated in an intelligent manner. Instead of randomly searching the image for interesting features, the focus of attention shifts in a direction determined by model knowledge and newly acquired information. In an island-driven approach where all of the objects are spatially connected, it becomes necessary to search only for the features that will instantiate the first item.

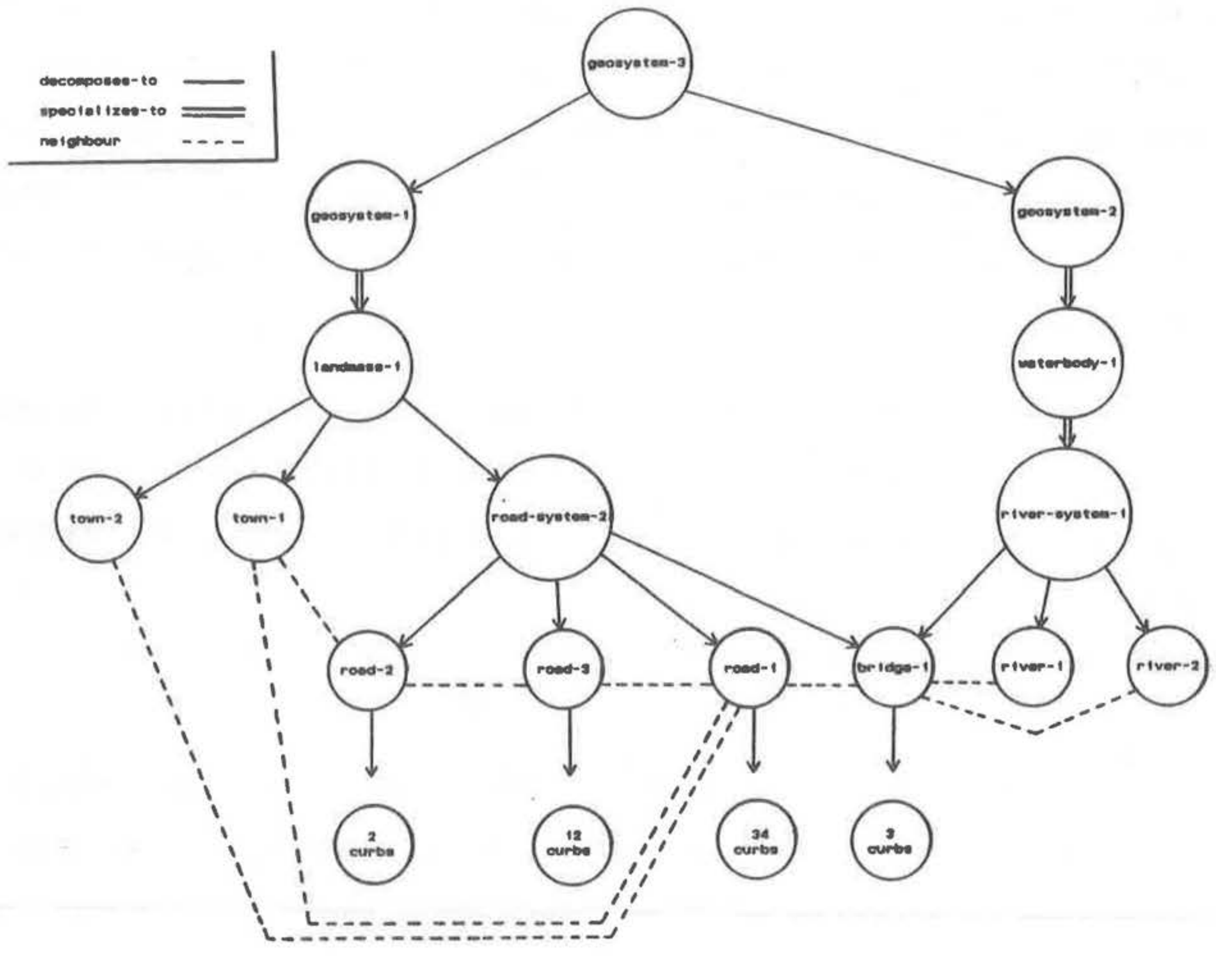


Figure 6.21 Ashcroft: Instance Hierarchy 3

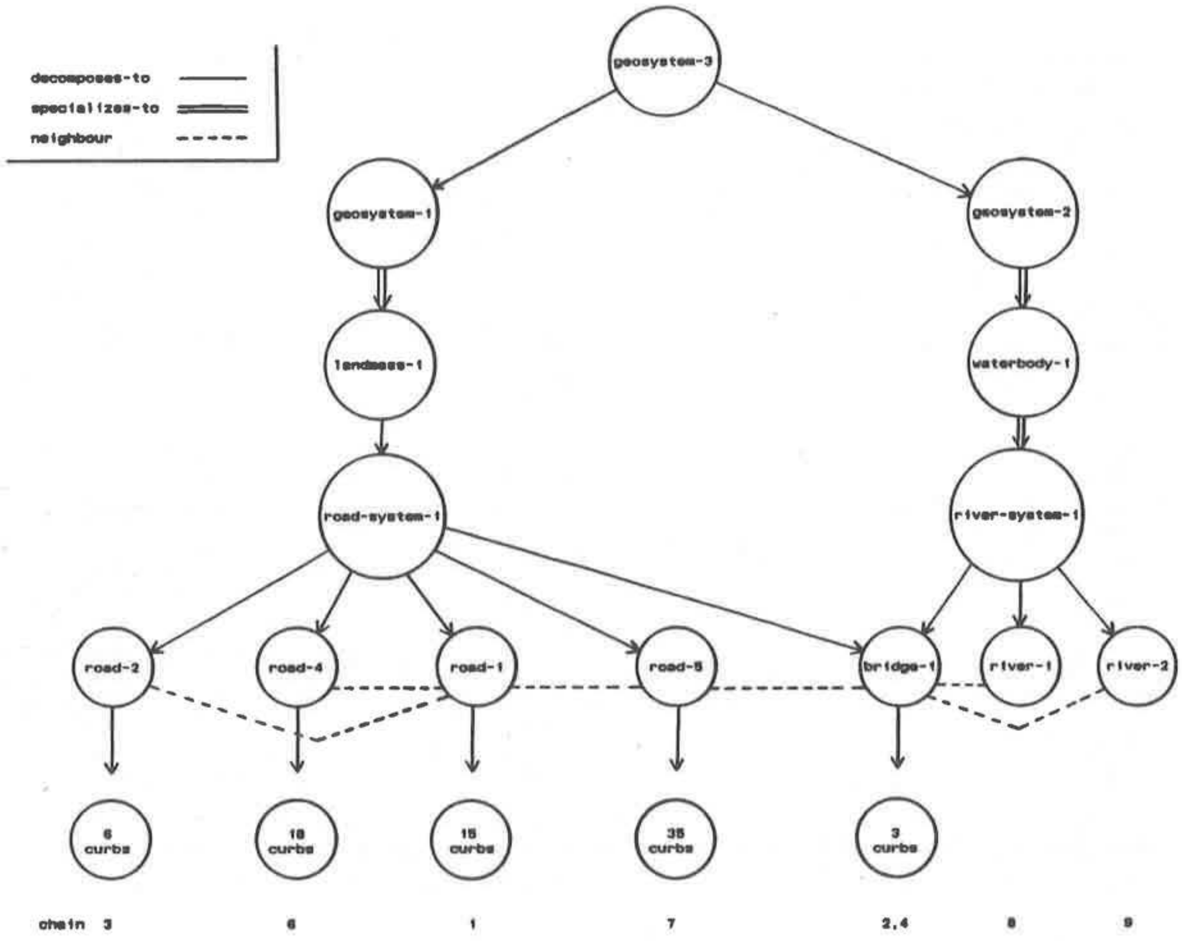


Figure 6.22 Houston: An Instance Hierarchy

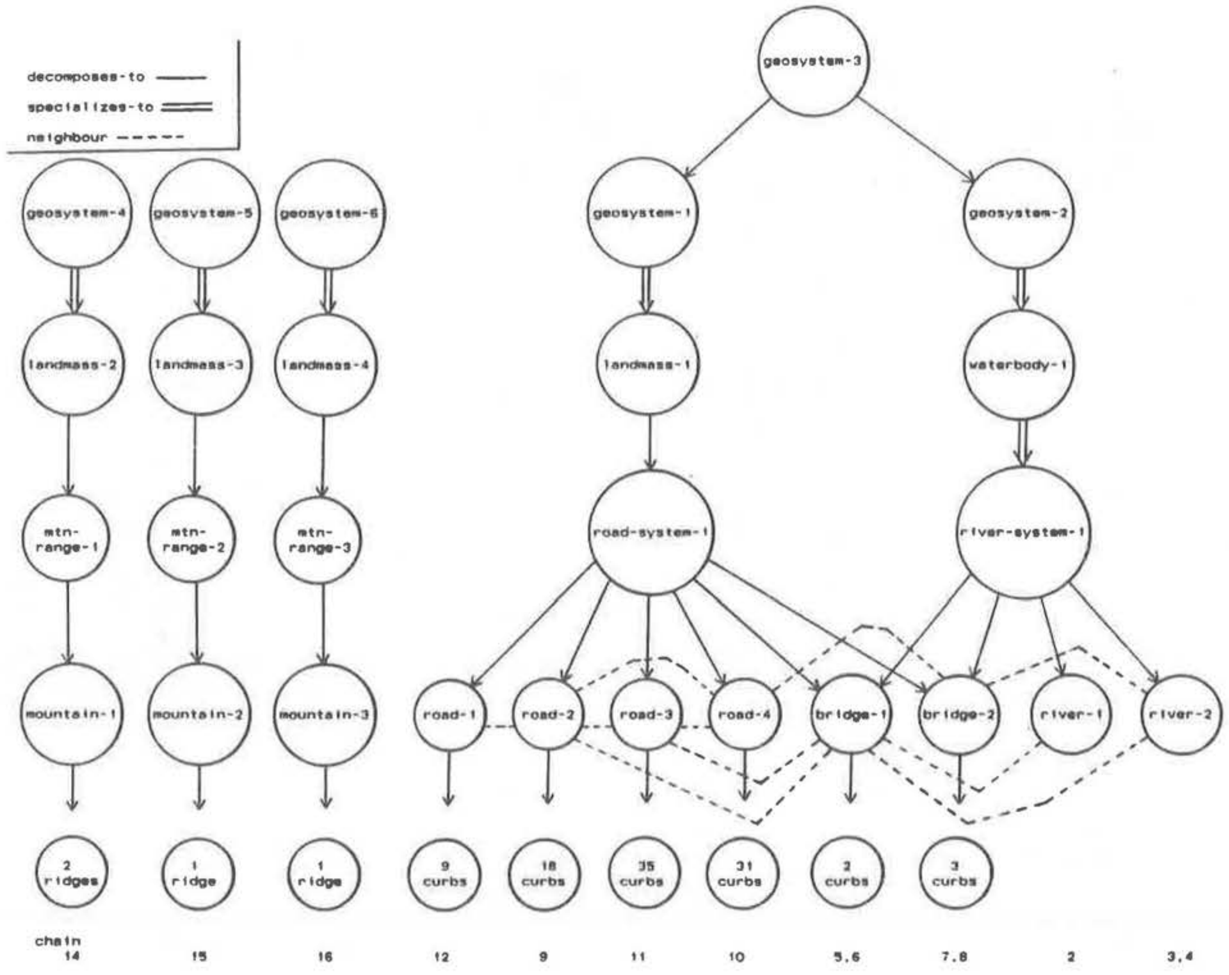


Figure 6.23 Spences Bridge: An Instance Hierarchy

When a sketch map IS is available, it is not even necessary to search (spatially) for the initial object. In fact, the organization of the sketch map schemata in their own semantic network might preclude the requirement of having to do it again for the image. No search would be necessary only if the chains in the sketch map were unambiguously instantiated and if the user was not interested in any entities that might appear in the image but not in the sketch map. While the latter will often be true, the former will seldom be.

So, to start the cycle of perception using sketch map information, an unambiguous instance is chosen. Bridges and towns are usually unique whereas roads, rivers, and mountains generally are not. Bridges also have the advantage of being part of both road and river systems and can provide much useful information about each. In particular, their attached procedures can determine image regions that are likely candidates for the associated road and river. After a suitable message is received, the top-down procedure associated with roads (and rivers) will search in the region for a chain in the sketch map. That chain probably has several interpretations. In this case, however, only the road instance will be used. This reduces search through the space of sketch map instances.

If the sketch map information source is not available, then model knowledge can be used to restrict the regions in the image that should be searched to find the initial object. In MISSEE, a simple heuristic has been used: calculate a range of suitable areas that a particular object might have (using the global value, scale) and then search them in order of decreasing size. Instantiation will again generate messages about likely interpretations for other regions. Since the sketch map information is not available, the less accurate (and less efficient) image-based routines will be directed to the appropriate regions or positions.

Table 6.8 shows the number of regions that were searched to instantiate the five bottom level schemata. The figures for

| | Ashcroft | Houston | Spences Bridge | Spences Bridge West | Spences Bridge East | Cranbrook |
|---|---|---|---|---|---|---|
| Total Number of Regions | 75 | 139 | 161. | 125 | 119 | 285 |
| With Sketchmap | | | | | | |
| Intended | | | | | | |
| #Relevant | 7 | 9 | 23 | 31 | 42 | 24 |
| #Extra | 6 | 12 | 4 | 1 | 1 | 20 |
| Unintended | 10 | 1 | 8 | 0 | 2 | 0 |
| Total | 23 | 22 | 35 | 32 | 45 | 44 |
| Without Sketchmap | | | | | | |
| Correct | | | | | | |
| #Relevant | 15 | 36 | 31 | 36 | 44 | 10 |
| #Extra | 17 | 39 | 15 | 27 | 13 | 9 |
| Incorrect | 7 | 7 | 51 | 9 | 1 | 3 |
| Total | 39 | 82 | 97 | 72 | 58 | 22 |

Table 6.8 The Number of Regions Searched

search with and without the use of a sketch map correspond to the tests that were used to compose Tables 6.1 to 6.7. Relevant regions are distinguished from extraneous ones in that they are the ones that are eventually used to fill in instance slots. As would be expected, the number of regions searched when the sketch map was employed was considerably smaller than without the sketch map. On the average, 22% of the regions were examined when the sketch map was used and 41% when it was not.

The advice given in messages is invariably useful in reducing search, because, even if all of the features that would enable instantiation are not present, the region is one that generally would have been searched anyway. For example, the procedures associated with roads indicate that neighbouring regions with the possible interpretation URBAN can be towns. The only other requirement for town is that its shape be roughly rectangular. If the region was a suitable size, it would have been searched anyway due to its intensity. Preserving negative (and positive) knowledge about regions and their attempted instantiation as objects prevents needless subsequent processing.

Search in the image is automatically reduced in three ways. 1) Positional information from the sketch map guides instantiation in the registered intensity image. 2) Model knowledge filters regions for further processing based on the size and average intensity of the region. 3) Model knowledge plus context (the information gained from the interpretation process) suggest likely interpretations for regions near the current focus of attention. Furthermore, possibilities for search are made explicit, giving the user control over their eventual use.

## 6.5. How User Interaction Can Influence Interpretation

In MISSEE, the user is able to do as much or as little as he/she desires to influence the progress of interpretation. By taking an active role, the user can guarantee that the results will be to his/her liking. Depending on how well the automatic system is performing, this may take more or less effort on his/her part.

At one extreme, the user may choose to let the system run automatically. It is not required that he/she draw a sketch map. In the current implementation, he/she must help train the intensity categorizer (Section 5.3.1.2.1) to assign possible classes to regions in the image, but there are ways to make this automatic, too. In terse mode, MISSEE will not interrupt its execution cycle but will continue until no more candidates remain[2]. MISSEE will have automatically produced a protocol and possibly images (if a flag was set indicating an appropriate output device was available). If that output is insufficient, then after the execution cycle is finished, the user may choose to examine the semantic network in greater detail.

[2]In the current system, that would be after every image region of suitable size and categorization has been tried with each of the five bottom level schemata--bridge, mountain, river, road, and town.

As indicated in Section 6.2, the results from this type of processing will be somewhat inaccurate, although they may well be adequate for some tasks. The user can help the situation significantly by drawing a sketch map of the scene over the image. Even better results will follow if the user takes a more active role in the interpretation process.

The user can provide three types of input to the system as was described in Section 4.4.1. In terms of improving the accuracy of the results, it may be necessary only to make minor changes in the parameters of messages on the possibilities list. More drastic measures would include adding or deleting messages. In extreme cases where scene objects have not been instantiated or non-existing entities have, it is possible (from Lisp) to modify the semantic network directly by creating and destroying instances. Thus, the user can ensure that the end results will be perfect. However, since there is no "natural" user interface, more extreme actions require the user to have more knowledge about the nature of messages and the schemata manipulation functions.

With regard to task priorities, the user can narrow the system's focus. If the user is only interested in objects of a certain type, such as roads or river systems, then he/she can direct MISSEE to look for only those objects (and their components and specializations) and to ignore irrelevant messages. In addition, the user can be very specific about where to find the items of interest. Then, he/she can examine specific parts of the resulting schemata to obtain the desired information about the objects.

The user also controls his/her environment so that the interaction fulfills his/her needs. Global input determines how much is included in the protocols, if and where graphic output is to be sent, and whether the user wants to have a chance to interact in each loop of the execution cycle. Such environment tailoring would become even more convenient if user models were utilized.

Appendix E contains a protocol for a sample session dealing with the Ashcroft image. The "user" has no specific priorities but is paying attention to the interpretation as it proceeds so that he can make slight modifications if necessary to ensure that the results will be accurate. Sketch map guidance is available and both top-down and bottom-up processing are illustrated. Also, messages of all sorts are generated: top-down, bottom-up, lateral, and user generated. While only the protocol is shown in Appendix E, some of the images produced would be similar to those found in Figures 5.15, 5.16, 5.21, 6.10, 6.11, 6.16, and 6.18. The resulting instances are found in Figure 6.21.

In summary, the user can influence the what, when, where, and how of interpretation to get the desired results. The amount of interaction can range from almost none to a step by step dialogue. The user would generally vary the amount of interaction depending on the accuracy of processing, his/her priorities, or both. This provides a very flexible adjunct to automatic processing.

## 6.6. Descriptive Adequacy

The descriptive adequacy of a vision system is determined by how well the descriptions of the domain serve to advance the solution of the problem at hand. In MISSEE, the task is not only the interpretation of intensity images but also the recovery of pertinent scene information. The schema-based system employed is definitely an asset in the pursuit of both of these goals.

Questions relating to descriptive adequacy were raised in Section 3.3. In response, the VALUEd and LINK types of the schemata make crucial facts explicit. These are the defining attributes of objects that are filled in via the attached procedures. Also, the slot values provide the source of knowledge after interpretation for gathering specific information about the elements of the scene. The attached procedures do not make explicit the methods by which instantiation takes place. This is an area where extensions might be made.

Facts are generally encoded in a uniform manner. The LINKs are maintained by the schemata manipulation system which enforces an explicit, uniform access to them. VALUEd types are specific to schemata but an attempt has been made to use standard names for commonly occurring attributes. These include "sketchmapitem", "regions", and "edges" to refer to the corresponding features from the information sources. Since bottom level schemata (bridges, curbs, mountains, ridges, rivers, roads, and towns) all interface to the intensity image by means of regions and/or edges, the method of description is consistent and uniform.

The clear distinction between general and particular knowledge enables the system to change as new information becomes available. Instances are hypothetical and can be created and destroyed based on the evidence. In contrast, stereotypic objects are static. New facts cause slots to be filled and an image-specific instance network to be built. This crucial distinction between the general and the specific is made at all levels in the hierarchies, not just at the leaf nodes.

Individual schemata are modular but not totally independent. Instantiation of bottom level schemata can take place independently from other schemata because they are self-contained, having both a procedural and a declarative part. However, higher level schemata that can not be instantiated directly from the IS's must have knowledge about their components or specializations so that control can be passed to an appropriate schema further down in the hierarchies. During instantiation, advisory messages can be sent (for efficiency purposes) laterally to neighbouring schemata which require knowledge of their attributes. In building the semantic network messages are sent up and down the hierarchies.

Even though schemata must "know" the names of the schemata they are sending messages to, modularity is maintained to a large extent because the messages all have a uniform format. The sender specifies only the type of procedure it wants to address, either top-down or bottom-up, and the system finds the appropriate procedure. Furthermore, the context parameters include information about what type of information they encode, such as "region 1629", "position (52 . 70)", "sketchmapitem *road-5", and rely on the receiver to extract the information it can utilize.

Previous sections have shown results that indicate that the descriptions in MISSEE are adequate for interpreting images. Moreover, they are adequate for combining facts from disparate IS's in a way that permits their effective use in the interpretation process. This is accomplished through the explicitness, uniformity, and modularity of the schemata and the clear distinction made between stereotypic and hypothetical knowledge.

## 6.7. Procedural Adequacy

By providing several types of attached procedures for each schema, MISSEE ensures that the appropriate method will be used for interpretation. Since the procedures are algorithms, they are flexible, easy to program (compared to a uniform processor operating in conjunction with a declarative data structure), and effective at image interpretation. Also, because the procedures are tightly coupled with the declarative portion of the schema (as opposed to a program and a separate data structure), greater modularity is maintained.

Schemata cooperate by means of the messages that are passed between them. It has been shown that this is an effective means of shifting the focus of attention to build the semantic network, interpret bottom level schemata, and provide context for subsequent processing.

Messages are uniform and include much of the information required to decode them. Schemata maintain modularity by relying on the receivers' of the message to determine the appropriate procedure to utilize. Thus a schema does not need to specify a specific bottom-up procedure for instantiation directly from an IS--i.e. whether to use sketch map guidance instead of the intensity image alone. Rather, the message is sent to the top-down procedure which takes the current context into account in determining which bottom-up procedure to use. More procedures might be added to a schema (perhaps dealing with new IS's) and only the receiver of the message would have to be modified to incorporate the new knowledge.

The control regime (the cycle of perception and the execution cycle) also maintains procedural adequacy. It allows the user to give and receive information at timely intervals so that he/she can influence processing. Furthermore, it ranks the interpretation possibilities for efficiency reasons, even though a complete search of all the regions, sketch map instances, and the like would be sufficient.

## 6.7.1. Solutions to Problems in Matching Data to Models

Section 3.2 described four problems in matching data to models in model-based vision systems. The adequacy of a vision system is partly determined by how well it solves those problems.

In inconsistent situations (more than one datum for a model), hypothetical models would be created to account for each data element, one-to-one. Additional information, especially from a different IS, may show that one instance does not fit the model well enough and cause it to be pruned from the network. If pruning does not reduce the number of instances to one, those that remain can be ranked by their confidence values which specify how closely the features from the IS's fit the model.

As an example, consider the case where one side of a bridge has been fixed (to edge-1) but there are two candidates for the other side (with edge-2 and edge-3 as data elements). Since the sides of a bridge should be parallel, the orientations of the edges can be used to choose between alternatives. If the orientation of either edge-2 or edge-3 was too dissimilar to that of edge-1, then the instance of which it was a part can be destroyed. Otherwise, the similarities can be used in the calculation of the confidence values so that the instances can be ordered in a meaningful way.

The ranking in terms of confidence values has another consequence with respect to the execution cycle. Any advice (in the form of messages) that is issued from the procedures attached to the two instances will also be ranked by the confidence values. This means, subject to user influence, that the most likely possibilities are examined first.

In the case where a model is unsatisfied (no corresponding data element) context can be a factor. If there is evidence that indicates the strong likelihood of the existence of a model, then the thresholds for accepting features can be lowered so that it is "easier" to instantiate the model. If crucial data elements (for example, the curbs of a road) can still not be found, then the hypothetical instance will be pruned. If the feature is not definitive, then the confidence value of the resulting instance would be lowered appropriately.

Ambiguous situations (more than one model for a data element) are handled in a manner similar to inconsistent ones. Each data element will cause the creation of a hypothetical instance. Other information will either result in some of the instances being pruned or will raise (or lower) their confidence values. For example, an ambiguous curb might be part of either a road or a bridge. The intensities of the neighbouring regions can be used to disambiguate the situation. Bridges are next to water which has a lower average intensity than urban regions which are generally found next to roads.

The final problem, incompleteness (no models to account for the data), has not been solved. Models and their attached procedures must be created by the user. Generic knowledge is static during the interpretation process[3]. To create new models as the result of interpretation would require the system to be capable of abstraction and learning. While this system might provide a convenient framework for such research, it was not attempted.

## 6.8. Robustness

The results described previously show that MISSEE degrades gracefully when it is unable to interpret an image completely (cf. Section 3.5). Moreover, the degradation of the results corresponds to the amount of information available from the IS's.

If the user maintains control over the interpretation, then MISSEE can produce perfect results. The robustness of the system ensures that as the user chooses to impart less information, the accuracy of the results diminishes gradually. If guidance from the sketch map is not available, the value of the results decreases further. However, even if the only source of information available is the digitized image, partial results will be produced.

[3]A user with knowledge of the schemata manipulation functions can create or modify schemata during interpretation by escaping into Lisp.

This range in the results and the robustness of the system has been illustrated in previous sections. Appendix E illustrates the case where the user ensures that the results are perfect (even though he only provided information in 6 out of the 36 cycles of execution). Section 6.1 indicated how good results were obtained using the guidance of a sketch map to help interpret an image (error rate, 9%). The results when using the digitized image alone were the subject of section 6.2 (where the error rate was found to be 24%).

The robustness of MISSEE is one of its main accomplishments. It supports the basic conjectures about multiple information sources. The more information and the more different the information that is utilized by the system, the better will be the results of the interpretation.

# CHAPTER 7

# Summary and Conclusions

## 7.1. A Summary of What is New and Interesting

1)Multiple Information Sources: The unifying conjecture of this work has been the possibility and usefulness of combining several, possibly disparate, types of input information sources to aid in the interpretation of visual images. Other researchers have used more than one information source, but this work raises the issue of how to effectively combine IS's to one of central importance.

One way to combine data that vary in kind and in their symbolic nature is to use an object-oriented data structure. MISSEE is a schema-based vision system that has been implemented to test the usefulness of the multiple information source conjectures. By using the information that is available to interpret images, it has proved effective in terms of adequacy and robustness. The design of MISSEE led to the development of several tools which are useful in reconciling different types of information to solve many of the problems in model-based vision.

There are two conjectures motivating this work. While there is general argreement about the usefulness of the first hypothesis (more is better), the second (the more different, the better) is harder to substantiate. Since there is no satisfactory definition of information in images, it is difficult to determine how disparate two information sources are. Nonetheless, intuitively a sketch map and an image are more different than two images recorded from different viewpoints. Information provided by a user is even more different than that found in images. While it is possible to extract from images alone all of the scene information that MISSEE is able to derive, it is maintained that the results indicate the advantage of combining the disparate sorts of information available from the various sources.

2)Schemata: A new variety of schemata has been implemented along with a system for manipulating them (MAIDS). These objects adapt many of the structuring notions of schemata developed for expert systems and natural language understanding programs to the specific needs of vision systems. In particular, vision systems require instances to be hypothetical at all levels of the hierarchies and distinct from static, generic objects. Also, multiple relationships between objects are necessary for the interpretation tasks that have been examined.

MAIDS schemata provide a consistent, uniform repository for information coming from different input IS's. The attached procedures are a flexible means of driving the interpretation process. Also, the construction of a semantic network linking all of the instances via the relations specializes to, decomposes to, and neighbours, allows for a method of control (the execution cycle) that reduces search, allows for user interaction, and promotes cooperation among the schemata.

3)Using Sketch Maps: Sketch map analysis has been used as a vision problem for some time. Sketch maps are quick and easy to draw even if they must be superimposed over an image, yet they contain much useful information about the underlying scene. This work is the first to use sketch map information to help interpret real imagery in a geographic domain.

4)Combining Edge and Region Data: Image analysis has usually concentrated on the derivation of image discontinuities (edges) or homogeneities (regions). Because in practice they are derived from different aspects of the image, these two representations provide different types of information. Following the multiple information sources conjecture, MISSEE is one of the first vision systems to utilize both edge and region data.

5)Mixed Control Strategies: Strictly linear models of control ineffectively utilize all of the knowledge that is available in a model-based vision system. A totally bottom-up system does not generally take advantage of model knowledge. A top-down system will not usually be able to use the context resulting from information gained from previous processing. MISSEE uses a mixed control strategy (both top-down and bottom-up) that combines with the structure of the semantic network so that attention is focussed where it can be most usefully applied among the cooperating schemata. The embodiment of this strategy is the execution cycle which also allows the user an opportunity to give and receive information at appropriate points although it is not necessary. If desired, the user can relate information of various types to the system and modify the list of possibilities for interpretation to reflect his/her own priorities.

The execution cycle distributes control within the semantic network. Messages directed at "bottom level" schemata can contain locations in either the sketch map or image. Advice about locality reduces the amount of search that must take place to discover the geographic entities of interest. This results in increased efficiency in contrast to the advances mentioned above which all concern the sufficiency or interpretive power of the system.

## 7.2. Open Issues

There are several aspects of the multiple information sources paradigm that have not been addressed. This research has dealt with one way of combining IS's--using schemata as both a data structure and object-oriented interpretation. There are several other candidates that have been used in AI that might also prove effective. For example, medical diagnosis systems that use production rules combine different types of test information in a non-perceptual domain.

When comparing different approaches to utilizing multiple information sources, a method of evaluation would be of considerable aid. If one could formalize the notions of the amount of information available in an IS, then it would be possible to determine the effectiveness of a method for extracting and using the information. Such a formalism would have to account for the fact that information from different IS's is not always cumulative and might be redundant or inconsistent.

With regard to the system that was developed, an open question exists concerning its expandability. Claims of modularity notwithstanding, it is unclear how easy and useful it would be in general to add either new objects or information sources. Ongoing experience with adding elements to the developing system was quite positive. However, only continued experimentation will indicate whether there are objects and IS's that are difficult to incorporate (perhaps clouds?) and whether increased size will lead to an overwhelming increase in complexity.

## 7.3. Future Directions

There are several extensions to the present system that would make it more effective, more efficient, or more modular.

1)Relaxing the Registration Constraint: In the current system, the sketch map is assumed to be approximately registered to the image. This gives spatial information regarding the location of objects. It should be possible to relax this constraint so that if the system discovers or is informed that the two IS's are not aligned, it will use relative instead of absolute spatial information (e.g. "There is a town to the left of a road that runs over a bridge").

2)Feedback to Mapsee2: Mapsee2 is run to completion on the sketch map and sends all of its results to MISSEE. As has been explained (Section 5.3.1), MISSEE can provide much information that is useful to help disambiguate the sketch map. Since Maya schemata, used in Mapsee2, are similar to MAIDS schemata, used in MISSEE, it should be possible for them to cooperate so that interpretation of the image and the sketch map can proceed simultaneously to their mutual benefit. However, the methods of control in Mapsee2 and MISSEE are substantially different, so considerable work would be required to coordinate them.

3)A Better User Interface: MISSEE's usefulness as a demonstration system would be improved with the addition of a natural language interface and a mechanism for graphical input, such as allowing the user to point to parts of the image or sketch map.

4)A More Declarative Schemata: It is possible to move some of the model knowledge from the attached procedures into the declarative part of the schemata. Moving the knowledge would make it more accessible, modifiable, and uniform, but would cause the instantiation method to be less flexible. There is a trend in other areas of AI towards more explicit, formal descriptions of objects. The framework of MISSEE would facilitate such a move.

5)Efficiently Finding Edges and Regions: In Section 5.3.1.2, it was pointed out that edge detection and region merging were both local operations that can be applied to selected portions of an image. In addition, to reduce search, MISSEE's control mechanism shifts the focus of attention in the image based on the current context (cf. Section 6.4). Thus one can confine the expensive operations of image analysis to specific parts of the image and only perform them when needed. This would require two changes to the current system: 1) one would have to refer to portions of the image in a new way (MISSEE generally uses the already calculated region numbers), and 2) boundary effects between neighbouring subimages would have to be considered.

## 7.4. Conclusions

Schema-based systems are capable of sufficiently describing the attributes of geographic entities for their interpretation, as well as directing the interpretation process itself. Exploiting information sources in a cooperative fashion gives results qualitatively better than those obtainable from interpreting the information sources separately.

## References

[1] Agin, G.J. and Binford, T.O. (1973), Computer Description of Curved Objects, Proc. Third International Joint Conference on Artificial Intelligence, Stanford, 629-640.

[2] Agin, G.J. and Duda, R.O. (1975), SRI Vision Research for Advanced Industrial Automation, Proc. Second USA-Japan Computer Conference, Tokyo, 113-117.

[3] Anstis, S.M. (1970), Phi Movement as a Subtraction Process, Vision Research 10(12), 1411-1430.

[4] Bajcsy, R. and Lieberman, L.I. (1974), Computer Description of Real Outdoor Scenes, Proc. Second International Joint Conference on Pattern Recognition, Copenhagen, 174-179.

[5] Bajcsy, R. and Joshi, A.K. (1978), A Partially Ordered World Model and Natural Outdoor Scenes, Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), Academic Press, New York, 263-270.

[6] Bajcsy, R. and Tavakoli, M. (1973), A Computer Recognition of Bridges, Islands, Rivers and Lakes from Satellite Pictures, Proc. Conf. on Machine Processing of Remotely Sensed Data, Lab. for Applications of Remote Sensing, Purdue, Indiana, 2A-54 to 2A-68.

[7] Bajcsy, R. and Tavakoli, M. (1976), Computer Recognition of Roads From Satellite Pictures, Proc. Second International Joint Conference on Pattern Recognition, Copenhagen, 190-194.

[8] Baker, H.H. and Binford, T.O. (1981), Depth From Edge and Intensity Based Stereo, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 631-636.

[9] Ballard, D.H. (1981), Parameter Networks: Towards a Theory of Low-Level Vision, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 1068-1078.

[10] Ballard, D.H. and Brown, C.M. (1982), Computer Vision, Prentice Hall, Englewood Cliffs, New Jersey.

[11] Ballard, D.H., Brown, C.M., and Feldman, J.A. (1978), An Approach to Knowledge-Directed Image Analysis, Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), Academic Press, New York, 271-282.

[12] Ballard, D.H. and Sabbah, D. (1981), On Shapes, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 607-612.

[13] Barr, A. and Feigenbaum, E.A. (eds.) (1981), The Handbook of Artificial Intelligence, Volume 1, William Kaufman, Los Altos, California.

[14] Barrow, H.G. and Tenenbaum, J.M. (1975), Representation and Use of Knowledge in Vision, Sigart Newsletter 52, 2-8.

[15] Barrow, H.G. and Tenenbaum, J.M. (1976), Msys: A System for Reasoning About Scenes, TN 121, AI Center, SRI.

[16] Barrow, H.G. and Tenenbaum, J.M. (1978), Recovering Intrinsic Scene Characteristics From Images, Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), Academic Press, New York, 3-26.

[17] Bartlett, F.C. (1932), Remembering, A Study in Experimental and Social Psychology, Cambridge University Press, Cambridge.

[18] Bobrow, D.G. and Winograd, T. (1977), An Overview of KRL, A Knowledge Representation Language, Cognitive Science 1, 3-46.

[19] Bolles, R.C. (1975), Verification Vision Within a Programmable Assembly System: An Introductory Discussion, AIM 275, AI Lab, Stanford.

[20] Bolles, R.C. (1979), Robust Feature Matching Through Maximal Cliques, Proc. Society of Photo-Optical Instrumentation Engineers 182, Imaging Applications for Automated Inspection and Assembly, 140-149.

[21] Bolles, R.C., Quam, L.H., Fischler, M.A., and Wolf, H.C. (1979), Automatic Determination of Image-to-Database Correspondences, Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, 73-78.

[22] Brachman, R. (1982), What "ISA" Is and Isn't, Proc. Fourth Conf. of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Canada, 212-221.

[23] Brice, C.R. and Fennema, C.L. (1970), Scene Analysis Using Regions, Artificial Intelligence 1(3), 205-226.

[24] Brooks, R.A. (1981a), Model-Based Three Dimensional Interpretations of Two Dimensional Images, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 619-624.

[25] Brooks, R.A. (1981b), Symbolic Reasoning Among 3-D Models and 2-D Images, Artificial Intelligence 17(1-3), 285-348.

[26] Brooks, R.A., Greiner, R., and Binford, T.O. (1979), The ACRONYM Model-Based Vision System, Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, 105-113.

[27] Brown, J.S. and Burton, R.R. (1975), Multiple Representations of Knowledge for Tutorial Reasoning, Representation and Understanding, D.G. Bobrow & A. Collins (eds.), Academic Press, New York, 311-349.

[28] Browse, R.A. (1982), Interpretation-Based Interaction Between Levels of Detail, Proc. Fourth Conf. of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Canada, 27-32.

[29] Clowes, M.B. (1971), On Seeing Things, Artificial Intelligence 2(1), 79-112.

[30] Collins, A.M. and Loftus, E.F.A. (1975), A Spreading-Activation Theory of Semantic Processing, Psychological Review 82(6), 407-428.

[31] Davis, L.S. (1975), A Survey of Edge Detection Techniques, Computer Graphics and Image Processing 4(3), 248-270.

[32] Davis, L.S. and Rosenfeld, A. (1981), Cooperating Processes for Low-level Vision: A Survey, Artificial Intelligence 17(1-3), 245-263.

[33] Davis, R. (1980a), Meta-Rules: Reasoning about Control, Artificial Intelligence 15, 179-222.

[34] Davis, R. (1980b), Report on the Workshop on Distributed AI, SIGART Newsletter 73, 42-52.

[35] Davis, R. (1982), Expert Systems: Where Are We? And Where Do We Go From Here?, AI Magazine 3(2), 3-22.

[36] Duda, R.O. and Hart, P.E. (1972), Use of the Hough Transformation to Detect Lines and Curves in Pictures, Communications of the ACM 15(1), 11-15.

[37] Dudani, S.A. and Luk, A.L. (1977), Locating Straight-Line Edge Segments on Outdoor Scenes, Proc. Pattern Recognition and Image Processing Conference, 367-377.

[38] Eigen, D.J., Fromm, F.R., and Northouse, R.A. (1974), Cluster Analysis Based on Dimensional Information with Applications to Feature Selection and Classification, Systems, Man, and Cybernetics SMC-4(3), 284-294.

[39] Erman, L.D. and Lesser, V.R. (1975), A Multi-Level Organization for Problem Solving Using Many, Diverse, Cooperating Sources of Knowledge, Proc. Fourth International Joint Conference on Artificial Intelligence, Tbilissi, 483-490.

[40] Erman, L.D., Hayes-Roth, F., Lesser, V.R., and Reddy, D.R. (1980), The HEARSAY-II Speech Understanding System: Integrating Knowledge To Resolve Uncertainty, Computing Surveys 12(2), 213-253.

[41] Falk, G. (1972), Interpretation of Line Data as a Three-Dimensional Scene, Artificial Intelligence 3(2), 101-144.

[42] Feigenbaum, E.A. (1977), Themes and Case Studies of Knowledge Engineering, Proc. Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass., 1014-1029.

[43] Fischler, M.A. and Elschlager, R.A. (1973), The Representation and Matching of Pictorial Structures, IEEE Trans. On Computers C-22(1), 67-92.

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

[45] Flinchbaugh, B.E. and Chandrasekaran, B. (1981), A Theory of Spatio-Temporal Aggregation for Vision, Artificial Intelligence 17(1-3), 387-407.

[46] Fowler, R.J. and Little, J.J. (1979), Automatic Extraction of Irregular Network Digital Terrain Models, Proc. SIGRAPH, Chicago, 199-207.

[47] Freuder, E.C. (1976), A Computer System for Visual Recognition Using Active Knowledge, AI-TR-345, AI Lab, MIT.

[48] Garvey, T.D., Lowrance, J.D., and Fischler, M.A. (1981), An Inference Technique for Integrating Knowledge From Disparate Sources, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 319-325.

[49] Gelertner, H. (1963), Realization of a Geometry-Theorem Proving Machine, Computers and Thought, E. Feigenbaum & J. Feldman (eds.), Mcgraw-Hill, New York, 134-152.

[50] Glicksman, J. (1982), A Schemata-Based System for Utilizing Cooperating Knowledge Sources in Computer Vision, Proc. Fourth Conf. of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Canada, 33-40.

[51] Goebel, R. (1977), Organizing Factual Knowledge in a Semantic Network, TR77-8, Dept. of Computer Science, U. of Alberta.

[52] Grimson, W.E.L. (1981), From Images to Surfaces: A Computational Study of the Human Early Visual System, MIT Press, Cambridge, Mass.

[53] Hall, E.L. (1979), Computer Image Processing and Recognition, Academic Press, New York.

[54] Hanson, A.R. and Riseman, E.M. (1978a), Segmentation of Natural Scenes, Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), Academic Press, New York, 129-164.

[55] Hanson, A.R. and Riseman, E.M. (1978b), VISIONS: A Computer System for Interpreting Scenes, Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), Academic Press, New York, 303-334.

[56] Havens, W.S. (1978), A Procedural Model of Recognition for Machine Perception, TR-78-3, Dept. of Computer Science, U.B.C.

[57] Havens, W.S. and Mackworth, A.K. (1980), Schemata-Based Understanding of Hand-Drawn Sketch Maps, Proc. Third Conf. of the Canadian Society for Computational Studies of Intelligence, Victoria, Canada, 172-178.

[58] Hayes, P., Ball, E., and Reddy, R. (1981), Breaking the Man-Machine Communication Barrier, Computer 14(3), 19-30.

[59] Hayes, P. and Reddy, R. (1979), Graceful Interaction in Man-Machine Communication, Proc. Sixth International Joint Conference on Aritficial Intelligence, Tokyo, 372-374.

[60] Hildreth, E.C. (1980), Implementation of a Theory of Edge Detection, AI-TR-579, AI Lab, MIT.

[61] Hewitt, C., Bishop, P., and Steiger, R. (1973), A Universal Modular ACTOR Formalism for Artificial Intelligence, Proc. Third International Joint Conference on Artificial Intelligence, Stanford, 235-245.

[62] Horn, B.K.P. (1975), Obtaining Shape From Shading Information, The Psychology of Computer Vision, P.H. Winston (ed.), McGraw-Hill, New York, 115-155.

[63] Horn, B.K.P. and Bachman, B.L. (1978), Using Synthetic Images to Register Real Images with Surface Models, Communications of the ACM 21(11), 914-924.

[64] Huffman, D.A. (1971), Impossible Objects as Nonsense Sentences, Machine Intelligence 6, B. Meltzer & D. Michie (eds.), American Elsevier, New York, 295-323.

[65] Jain, R. and Haynes, S. (1982), Imprecision in Computer Vision, Computer 15(8), 39-48.

[66] Kanade, T. (1980), Region Segmentation: Signal vs. Semantics, Computer Graphics and Image Processing 13, 279-297.

[67] Kanade, T. (1981), Recovery of the Three-Dimensional Shape of an Object from a Single View, Artificial Intelligence 17(1-3), 409-460.

[68] Kelly, M.D. (1971), Edge Detection in Pictures by Computer Using Planning, Machine Intelligence 6, B. Meltzer & D. Michie (eds.), American Elsevier, New York, 397-409.

[69] Kender, J.R. (1980), Shape from Texture, CMU-CS-81-102, Dept. of Computer Science, CMU.

[70] Koomen, J.A.G.M. (1980), The Interlisp Virtual Machine: A Study of its Design and its Implementation as Multilisp, M.Sc. dissertation, Dept. of Computer Science, U.B.C.

[71] Kuhn, T.S. (1970), The Structure of Scientific Revolutions, edition 2, University of Chicago Press, Chicago.

[72] Kuipers, B.J. (1977), Representing Knowledge of Large-Scale Space, AI-TR-418, AI Lab, MIT .

[73] Leboucher, G. and Lowitz, G.E. (1978), What a Histogram Can Really Tell the Classifier, Pattern Recognition 10(4), 351-357.

[74] Levesque, H. and Mylopoulos, J. (1979), A Procedural Semantics for Semantic Networks, Associative Networks: The Representation and Use of Knowledge by Computer, N.V. Findler (ed.), Academic Press, New York, 93-120.

[75] Levine, M.D. and Shaheen, S.I. (1981), A Modular Computer Vision System for Picture Segmentation and Interpretation, Pattern Analysis and Machine Intelligence PAMI-3(5), 540-556.

[76] Lillesand, T.M. and Kiefer, R.W. (1979), Remote Sensing and Image Interpretation, John Wiley and Sons, New York.

[77] Little, J.J. (1982), Automatic Registration of Landsat MSS Images to Digital Elevation Models, Proc. Workshop on Computer Vision: Representation and Control, Rindge, New Hampshire, 178-184.

[78] Mackworth, A.K. (1976), Model-Driven Interpretation in Intelligent Vision Systems, Perception 5, 349-370.

[79] Mackworth, A.K. (1977a), Consistency in Networks of Relations, Artificial Intelligence 8(1), 99-118.

[80] Mackworth, A.K. (1977b), How to See a Simple World: an Exegesis of Some Computer Programs for Scene Analysis, Machine Intelligence 8, E.W. Elcock & D. Michie (eds.), John Wiley, New York, 510-540.

[81] Mackworth, A.K. (1977c), On Reading Sketch Maps, Proc. Fifth International Joint Conference on Artificial Intelligence, Cambridge, Mass., 598-606.

[82] Mackworth, A.K. (1978), Vision Research Strategy: Black Magic, Metaphors, Mechanisms, Miniworlds and Maps, Computer Vision Systems, A.R. Hanson & E.M. Riseman (eds.), Academic Press, New York, 53-61.

[83] Mackworth, A.K. and Havens, W.S. (1981), Structuring Domain Knowledge for Visual Perception, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 625-627.

[84] Marr, D. (1976), Early Processing of Visual Information, Phil. Trans. Royal Society of London 275B(942), 483-524.

[85] Marr, D. (1982), Vision: A Computational Investigation into the Human Representation and Processing of Visual Information, W.H. Freeman, San Francisco.

[86] Marr, D. and Hildreth, E.C. (1980), Theory of Edge Detection, Proc. Royal Society of London B(207), 187-217.

[87] Marr, D. and Nishihara, H.K. (1978), Representation and Recognition of the Spatial Organization of Three Dimensional Structure, Proc. Royal Society of London B(200), 269-294.

[88] Minsky, M. (1975), A Framework for Representing Knowledge, The Psychology of Computer Vision, P.H. Winston (ed.), McGraw-Hill, New York, 211-277.

[89] Mulder, J.A. (1979), Representation and Control in a Program that Understands Line Sketches of a House, M.Sc. dissertation, Dept. of Computer Science, U.B.C.

[90] Nagao, M., Matsuyama, T., and Ikeda, Y. (1978), Region Extraction and Shape Analysis of Aerial Photographs, Proc. Fourth International Joint Conference on Pattern Recognition, Kyoto, 620-628.

[91] Nagao, M., Matsuyama, T., and Mori, H. (1979), Structural Analysis of Complex Aerial Photographs, Proc. Sixth International Joint Conference on Aritficial Intelligence, Tokyo, 610-616.

[92] Nazif, A.M. and Levine, M.D. (1982), Setting Strategy in a Rule Based Vision System, Proc. Fourth Conf. of the Canadian Society for Computational Studies of Intelligence, Saskatoon, Canada, 52-59.

[93] Neisser, U. (1967), Cognitive Psychology, Meredith, New York.

[94] Neisser, U. (1976), Cognition and Reality: Principles and Implications of Cognitive Psychology, W.H. Freeman, San Francisco.

[95] Nevatia, R. and Price, K.E. (1978), Locating Structures in Aerial Images, Proc. Fourth International Joint Conference on Pattern Recognition, Kyoto, 686-690.

[96] Nevatia, R. and Price, K.E. (1982), Locating Structures in Aerial Images, Pattern Analysis and Machine Intelligence PAMI-4(5), 476-484.

[97] Nii, H.P., Feigenbaum, E.A., Anton, J.J., and Rockmore, A.J. (1982), Signal-to-Symbol Transformation: HASP/SIAP Case Study, AI Magazine 3(2), 23-35.

[98] Nilsson, N. J. (1971), Problem-Solving Methods in Artificial Intelligence, McGraw-Hill, New York.

[99] Nilsson, N. J. (1980), Principles of Artificial Intelligence, Tioga, Palo Alto, California.

[100] O'Gorman, F. and Clowes, M.B. (1973), Finding Picture Edges Through Collinearity of Feature Points, Proc. Third International Joint Conference on Artificial Intelligence, Stanford, 543-555.

[101] Ohlander, R.B. (1975), Analysis of Natural Scenes, Ph.D. dissertation, Dept. of Computer Science, CMU.

[102] Ohta, Y., Kanade, T., and Sakai, T. (1978), An Analysis System for Scenes Containing Objects with Substructures, Proc. Fourth International Joint Conference on Pattern Recognition, Kyoto, 752-754.

[103] Palmer, S.E. (1975), Visual Perception and World Knowledge: Notes on a Model of Sensory-Cognitive Interaction, Explorations in Cognition, D.A. Norman & D.E. Rumelhart (eds.), W.H. Freeman, San Francisco, 279-307.

[104] Postaire, J.-G. and Vasseur, C.P.A. (1981), An Approximate Solution to Normal Mixture Identification with Application to Unsupervised Pattern Classification, Pattern Analysis and Machine Intelligence PAMI-3(2), 163-179.

[105] Ramer, U. (1972), An Iterative Procedure for the Polygonal Approximation of Planar Curves, Computer Graphics and Image Processing 1(3), 244-256.

[106] Rich, E. (1979a), Building and Exploiting User Models, Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, 720-722.

[107] Rich, E. (1979b), Building and Exploiting User Models, CMU-CS-79-119, Dept. of Computer Science, CMU.

[108] Roberts, L.G. (1965), Machine Perception of Three Dimensional Solids, Optical and Electro-optical Information Processing, J.T. Tippett, D. Berkowitz, L. Clapp, C. Koester, & A. Vanderburgh (eds.), MIT Press, Cambridge, Mass., 159-197.

[109] Roberts, R.B. and Goldstein, I.P. (1977a), The FRL Manual, AIM 409, AI Lab, MIT.

[110] Roberts, R.B. and Goldstein, I.P. (1977b), The FRL Primer, AIM 408, AI Lab, MIT.

[111] Robson, D. and Goldberg, A. (1981), The Smalltalk-80 System, Byte 6(8), 36-47.

[112] Rosenberg, S.T. (1977), Frame-Based Text Processing, AIM 431, AI Lab, MIT.

[113] Rosenthal, D. and Bajcsy, R. (1978), Conceptual and Visual Focussing in the Recognition Process as Induced by Queries, Proc. Fourth International Joint Conference on Pattern Recognition, Kyoto, 417-420.

[114] Russell, D.M. (1979), Where Do I Look Now?: Modelling and Inferring Object Locations by Constraints, Proc. IEEE Comp. Society Conf. on Pattern Recognition and Image Processing, Chicago, 175-181.

[115] Sakai, T., Kanade, T., and Ohta, Y. (1976), Model-Based Interpretation of Outdoor Scenes, Proc. Third International Joint Conference on Pattern Recognition, Coronada, California, 581-585.

[116] Schacter, B.J., Davis, L.S., and Rosenfeld, A. (1976), Scene Segmentation by Cluster Detection in Color Spaces, SIGART Newsletter 58, 16-17.

[117] Scott, D.W. (1979), On Optimal and Data-based Histograms, Biometrika 66(3), 605-610.

[118] Selfridge, P.G. and Sloan, K.R. (1981), Reasoning About Images: Using Meta-Knowledge in Aerial Image Understanding, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 755-757.

[119] Shirai, Y. (1975), Analyzing Intensity Arrays Using Knowledge About Scenes, The Psychology of Computer Vision, P.H. Winston (ed.), McGraw-Hill, New York, 93-113.

[120] Shortliffe, E.H. and Buchanan, B.G. (1975), A Model of Inexact Reasoning in Medicine, Mathematical Biosciences 23, 351-379.

[121] Sloan, K.R. (1982), Analysis of "Dot Product Space" Shape Description, Pattern Analysis and Machine Intelligence PAMI-4(1), 87-90.

[122] Sloan, K.R. and Bajcsy, R. (1977), World Model Driven Recognition of Natural Scenes, Dept. of Computer and Information Science, Moore School of E.E., U. of Pennsylvania, Philadelphia.

[123] Smith, R.G. and Friedland, P. (1980), UNIT Package User's Guide, Technical Memorandum 80/L, DREA.

[124] Stefik, M.J. (1979), An Examination of a Frame-Structured Representation System, Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, 845-852.

[125] Stefik, M.J. (1980), Planning with Constraints, STAN-CS-80-784, Department of Computer Science, Stanford.

[126] Stoch, L. (1981), Interactive Polygon Filling on a Raster Graphic Display, M.Sc. dissertation, Dept. of Computer Science, U.B.C.

[127] Stockman, G. (1978), Toward Automatic Extraction of Cartographic Features, ETL-0153, L.N.K. Corporation, Silver Spring, Maryland.

[128] Tavakoli, M. and Rosenfeld, A. (1982), Building and Road Extraction from Aerial Photographs, Systems, Man, and Cybernetics SMC-12(1), 84-91.

[129] Tenenbaum, J.M. (1973), On Locating Objects by Their Distinguishing Features in Multisensory Images, TN 84, AI Center, SRI.

[130] Tenenbaum, J.M. and Barrow, H.G. (1977), Experiments in Interpretation-Guided Segmentation, Artificial Intelligence 8(3), 241-274.

[131] Tenenbaum, J.M., Fischler, M.A., and Wolf, H.C. (1978), A Scene-Analysis Approach to Remote Sensing, TN 173, AI Center, SRI.

[132] Thorndyke, P.W. (1979), Heuristics for Knowledge Acquisition from Maps, Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, 880-883.

[133] Tomita, F. (1981), Hierarchical Description of Textures, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 728-733.

[134] Ullman, S. (1978), The Interpretation of Visual Motion, MIT Press, Cambridge, Mass.

[135] Wesley, L.P. and Hanson, A.R. (1982), The Use of an Evidential-Based Model for Representing Knowledge and Reasoning About Images in the VISIONS System, Proc. Workshop on Computer Vision: Representation and Control, Rindge, New Hampshire, 14-25.

[136] Weymouth, T.E. (1981), Experiments in Knowledge-Driven Interpretation of Natural Scenes, Proc. Seventh International Joint Conference on Artificial Intelligence, Vancouver, 628-630.

[137] Winston, P.H. (1972), The MIT Robot, Machine Intelligence 7, B. Meltzer & D. Michie (eds.), American Elsevier, New York, 431-463.

[138] Winston, P.H. (1977), Artificial Intelligence, Addison-Wesley, Reading, Mass..

[139] Witkin, A.P. (1981), Recovering Surface Shape and Orientation from Texture, Artificial Intelligence 17(1-3), 17-45 .

[140] Woodham, R.J. (1980a), Photometric Method for Determining Surface Orientation from Multiple Images, Optical Engineering 19, 139-144.

[141] Woodham, R.J. (1980b), Using Digital Terrain Data to Model Image Formation in Remote Sensing, Proc. Society of Photo-Optical Instrumentation Engineers 238, 361-369.

[142] Woodham, R.J. (1981), Analyzing Images of Curved Surfaces, Artificial Intelligence 17(1-3), 117-140.

[143] Yachida, M., Ikeda, M., and Tsuji, S. (1979), A Knowledge Directed Line Finder for Analysis of Complex Scenes, Proc. Sixth International Joint Conference on Artificial Intelligence, Tokyo, 984-991.

[144] Yakimovsky, Y. and Feldman, J.A. (1973), A Semantics-Based Decision Theory Region Analyzer, Proc. Third International Joint Conference on Artificial Intelligence, Stanford, 580-588.

[145] Zucker, S.W., Rosenfeld, A., and Davis, L.S. (1975), General Purpose Models: Expectations about the Unexpected, Proc. Fourth International Joint Conference on Artificial Intelligence, Tbilisi, 716-721.

# APPENDIX A

## The MAIDS System

### Schemata

Schemata are stored on the property list of the atom whose pname is the name of the object or instance. There are two types of schemata:

OBJECTS--stereotypic or general knowledge:

They have %object% as the first property on their property list.

INSTANCES--particular knowledge created as part of interpretation:

They are of the form objectname-number where the numbers are incremented from 1 as instances are created. They have %instance% as the first property on the property list.

### The Four Parts of a Schemata

```
VALUEd type = attribute-name (%VALUE S-expresseion
                             ((%CONFIDENCE  . number)
                              (%DEFAULT     . S-expression)
                              (%IF-ADDED    . form)
                              (%IF-MODIFIED . form)
                              (%IF-REMOVED  . form)
                              (%REQUIRED    . predicate)))

LINK type   = link-name {schema | (schema1 schema2 ...)}

CONFIDENCE  = CONFIDENCE number
```

CONF-ALG form

PROCEDUREs -> procedures to be evaluated when appropriate messages are received

## Global Variables

The global variable %link_types records all the known relationships and their inverses. It is of the form

```
%link_types = ((ako . specializes-to)(specializes-to . ako)
               (apo . decomposes-to) (decomposes-to  . apo)
               (aio . instances)     (instances      . aio)
               . . .)
```

%schemata: a list of all the generic (stereotypic) schemata available in the loaded system

%instances: a list of all the particular instances available in the loaded system

## Local Variables

In the procedures attached to CONFIDENCE values (CONF-ALG) and VALUEd types (IF-ADDED, IF-MODIFIED, IF-REMOVED, IF-NEEDED, REQUIRED), the name of the current schema can be found in the variable %name.
Furthermore, in VALUEd type attached procedures, the value that has just been added, modified, etc. is located in the variable %val.

## Conventions

In the following description of functions, all arguments are evaluated unless explicitly noted, optional arguments are enclosed in "[]"'s, and the following suffixes are generally used:

| | |
|---|---|
| A | any Attribute |
| V | VALUEd type attributes |
| L | LINK attributes |
| I | Instances |
| O | Objects |

## Schemata Manipulation Functions

1.(add-link-type link)

Returns: link

Side effect: Adds a link and its inverse to the global variable %link_types: a list of all the known link types. The user will be queried for the name of the inverse but it is not necessary for one to be supplied.

2.(add-link-inverse link inverse)

Returns: nil

Side effect: Makes inverse the inverse of link in the global variable %link_types. If link and inverse do not exist in %link_types, they are both added.

3.(ifneed schema attribute)

Returns: the result of evaling the %if-needed clause of attribute (a VALUEd type) in schema.

4.(saddl schema1 link schema2)

Returns: schema2

Side effects: Adds a LINK from schema1 to schema2. In addition, the inverse of the LINK is added from schema2 to schema1. Link must be known to %link_types.

5.(sanylink? schema1 schema2)

Returns: If there is a path between schema1 and schema2 along any type of LINK it returns a list of all the link types used in the path (of the form (link1 link2 link3 ...)). If there is no path, it returns nil.

6.(sattr schema type)

Where: Type is either nil (return all attributes) or a list of one or more or the following types to be returned: valued, link, procedure, or other.

Returns: A list of the attributes of schema depending on type.

7.(sattrtype schema attribute)

Returns: the type of attribute in schema. Type will be one of valued, link, procedure, or other.

8.(sconsist)

Returns: nil

Side effect: Makes a data base consistent by adding and removing appropriate LINKs and objects. After it is finished, all LINK types will point to existing objects and their inverses will also be in place.

9.(screate [name])

Where: name is the unevaluated name of the schema that will be created. If name is not provided the user will be queried for it. In addition, the user will be queried for the various parts of the schema and can input any attributes that are desired.

Returns: The newly created object.

Side effect: The property list of the named schemata will be modified to include all the attributes that it is initially endowed with.

10.(serasei instance [if-rem? [splice?]])

Where: if-rem? and splice? are either non-nil or nil.

Returns: the name of the destroyed instance.

Side effect: Destroys instance as a schema (removes the attributes from its property list) and removes all the LINKs to it. If if-rem? is non-nil, then all the existing if-removed clauses on VALUEd types are evaled. If splice? is non-nil, then the LINKs from adjacent schemata are spliced around instance (see splice).

11.(seraseo object splice?)

Where: splice? is non-nil or nil.

Returns: the name of the destroyed object.

Side effect: Destroys the stereotype object. Since this is not likely to be done very often, the user is prompted to make sure he/she really wants to do it. If splice? is non-nil, then the schemata adjacent to object have their links spliced around it (see splice).

12.(serrn form culprit)

Where: n is a number from 1 to 12.

Side effect: Error message n is printed using the function form and value culprit. Debug is then called on form.

13.(sgeta schema attribute [inherit [allorone [links]]])

Where: inherit is one of "y" or "n"; allorone is one of "a" or "o"; and links is a list of LINKs.

Returns: The value of attribute in schema. If the attribute in schema has a value, then it is returned. If there is no attribute in schema, then property inheritance will be used (if inherit = y as opposed to n). Inheritance will be sought up the LINKs found in the list "links". Depending on the value of allorone, either the first value (from a breadth-first search) will be returned (allorone = o) and the global variable %wherefrom will be set to the schema where the attribute was found, or all the values and their schemata will be returned in a list e,g. ((val1 . sch1)(val2 . sch2)) (if allorone = a).

Note: The default values are inherit=n, allorone=o, links=(ako).

14.(sgetc schema [attribute])

Returns: the confidence value of a schema (if attribute = nil or "confidence") or of a VALUEd type attribute.

15.(sgetl schema link)

Returns: the schema (or schemata) "linked" to schema via link.

16.(sgetv schema attribute inherit type allorone links1 links2)

Where: inherit is one of "y" or "n"; type is one of "v", "vd", or "d"; allorone is one of "a" or "o"; and links1 and links2 are lists of LINKs.

Returns: The value (or default) of attribute (which must be a VALUEd type) in schema. If there is no attribute in schema, then property inheritance will be used (if inherit = y as opposed to n). Inheritance will be sought for either a value only (type = v, using links1), a value or a default at every step (type = vd, using links1), or the whole structure will be searched for a value (using links1) and if none is found the whole structure will again be searched for a default (using links2) (type = d). Depending on the value of allorone, either the first value (from a breadth-first search) will be returned (allorone = o) and the global variable %wherefrom will be set to the schema where the attribute was found, or all the values and their schemata will be returned in a list e.g. ((val1 . sch1)(val2 . sch2)) (if allorone = a). If a default as opposed to a value is returned, then the above description holds for defaults plus the global variable %default is set to t.

Note: defaults are inherit = y, type = vd, allorone = o, links1 = (ako), and links2 = (ako).

17.(slink? schema1 schema2 link)

Returns: If there is a path from schema1 to schema2 along link, then the minimum length of the path, else nil. If link is nil, then sanylink? is called which looks for a path using any type of link.

18.(smerge schema1 schema2 doconfalgorithm?)

Where: doconfalgorithm? is non-nil or nil.

Returns: the priority queue.

Side effect: Combines the slots of schema2 into those of schema1, i.e., for VALUEd types, if the attribute in schema2 has a non-nil value and the one in schema1 does not, put the value in schema1's slot. For LINK types, take the union of the pointers. If doconfalgorithm? is true, evaluate the CONFIDENCE algorithm on the resulting schema. Also, all references to schema2 in previous messages in the priority queue are changed to schema1.

19.(snewi object)

Returns: the new instance.

Side effect: Creates a new instance of the stereotype object. It will have the name object-n, where n is the next integer in sequence (starting from 1).

20.(splice schema link [inverse])

Returns: nil

Side effect: Removes the LINKs from schema to the other nodes attached by link and splices the LINKs around it.
Thus:

```
  up                          up
   |                           |
   |inverse                    |inverse
   |                           |
schema          goes->         |
   |                           |
   |link          to           |link
   |                           |
 down                        down
```

21.(sprint schemaname)

Returns: nil

Side effect: Pretty prints a schema.

22.(sprintn schema link)

Returns: nil

Side effect: Pretty prints the tree structure under schema using link.

23.(sputa schema attribute [value])

Returns: the value of "value".

Side effect: Puts a new definition of attribute in schema and places "value" as its initial value.

24.(sputc schema [attribute [cval [spread? [skip?]]]])

Where: cval is a number, and spread? and skip? are either non-nil or nil.

Returns: the old confidence value.

Side effect: Changes the CONFIDENCE value of attribute in schema to cval. If attribute = nil or "confidence" the value of CONFIDENCE at the schema level is changed; otherwise the VALUEd type attribute has its CONFIDENCE changed. If spread? is non-NIL then the effects of the change (if there are any) are spread by re-evaling all the conf-algorithms up the apo and ako hierarchies. If spread? is non-NIL and skip? is, then the CONF-ALG is evaluated for schema (i.e. the value of cval is ignored).

25.(sputcl schema attribute value oldconfidence newconfidence spread?)

Where: spread? is either non-nil or nil.

Returns: the old confidence value.

Side effect: Replaces oldconfidence in the list of confidence values of attribute in schema with newconfidence. If spread? is non-NIL then the effects of the change (if there are any) are spread by re-evaling all the conf-algorithms up the apo and ako hierarchies.

26.(sputv schema attribute value)

Returns: value.

Side effect: Puts a new value into the %value slot of attribute in schema. Depending on the previous and new values, certain demons may respond.

| previous value | new value | | |
|---|---|---|---|
| | ~nil | %required | is checked |
| nil | nil | | |
| nil | ~nil | %if-added | is evaled |
| ~nil | ~nil | %if-modified | is evaled |
| ~nil | nil | %if-removed | is evaled |

27.(sremovea schema attribute [if-remclause?])

Where: if-remclause? is either non-nil or nil.

Returns: the value of the removed attribute.

Side effect: Removes the definition of attribute from schema. If the if-remclause? is non-nil and there is a non-nil value, then the if-removed form (if it exists) is evaled.

28.(sremoveal schema link)

Returns: the value of the schema(ta) pointed to from "schema" by link.

Side effect: Removes all the LINKs attaching schema to other schemata via link.

29.(sremovel schema1 link schema2)

Returns: schema2

Side effect: Removes the link from schema1 to schema2 and the inverse link from schema2 to schema1. If schema2 is nil, then sremoveal is called which removes all the nodes attached to schema1 by link.

30.(sremovev schema attribute)

Returns: nil.

Side effect: Removes the %value from attribute (a VALUEd type) in schema (by setting it to nil).

31.(srestore [file])

Where: file is not evalled.

Returns: nil.

Side effect: Restores the stereotype schemata from file. if file is nil, then the default file "saved.d" is used.

32.(ssave [file])

Where: file is not evalled.

Returns: file.

Side effect: Saves all the known stereotype schemata on file. If file is nil, then the default file "saved.d" is used.

33.(ssprintn schema)

Returns: nil

Side effect: Pretty prints the tree structure under schema following all links.

34.(vcheck schema attribute value)

Returns: the result of evalling the %required clause on value (if it exists) of attribute (a VALUEd type) in schema.

# APPENDIX B

## Edge Detection

An edge detector has been implemented based on the Marr-Hildreth theory of edge detection (Marr and Hildreth, 1980; Hildreth, 1980). Edges are located at the zero crossings of the Laplacian of the Gaussian function convolved with the image. This appendix describes specific implementation details of how edges and edge information are calculated.

A mask 31 pixels square, centred on the origin, is generated from the function

$$m(x,y) = (x^2 + y^2 - 2\sigma^2) * \exp(-(x^2 + y^2)/2\sigma^2)$$

and then scaled so that the value of $m(0,i) = 0.5$ where $i$ is the largest integer $\leq 15$ such that $m(0,i)$ is non-zero. In contrast to the Marr-Hildreth theory, only one $\sigma$ value is used, and it is chosen to produce high resolution edges. Arithmetic is carried out using floating point arithmetic.

The mask is convolved with the digitized images (128 by 128 by 256 grey scale values). Zero crossings are found by tracing out curves in the convolved image where the values change signs. This produces continuous curves that are either closed or run off the edge of the image.

Since these long curves generally include several semantically distinct objects, they are broken into shorter straight line segments by a method called generalization (Ramer, 1972; Little, 1982). Generalization is a recursive process wherein a line is drawn between the endpoints of a curve and the point on the curve furthest from the line is found. If the distance from the line to the point is greater than a threshold (a value of 1 pixel was used) then the line is subdivided at the point and the two new segments are generalized in turn. Recursion stops when the distance of all the points from the line is less than the threshold, guaranteeing that the deviation is less than that value. This produces a number of connected, "straight" segments.

From this process a line image is returned that corresponds to the original image. Each pixel contains either 0, meaning that there is no edge segment at that point, or a number identifying the edge segment it is a part of. A separate data structure contains information for each edge segment. This includes location, length, orientation (from 0 to 360 degrees since the value is also used to indicate that the region to the right of the segment is brighter than the one on the left), and contrast.

Contrast is the first directional derivative across the edge segment (not the slope of the third as was suggested by Marr and Hildreth). It is calculated by moving along the points in the intensity image that correspond to those of the line image. For each point, the difference is calculated between the pair of pixel values one pixel away from the point and perpendicular to the orientation of the segment. The average of the absolute values of all the differences is the contrast (or strength) of the segment.

This information forms a type of raw primal sketch. It is (one of) the intermediate representations accessed by higher level functions.

## Appendix B

# APPENDIX C

## Region Merging

Regions in the intensity image are merged using an implementation of Freuder's affinity method (Freuder, 1976). The general notion is to merge regions that have the greatest affinity towards each other, based mainly on their intensities.

Freuder started by forming initial regions 25 units on a side (out of a 300 by 350 pixel image). Since some significant features in the images that have been used can be only a few pixels in extent, a different initialization was performed. Each pixel in the image that is not part of a region is chosen as a seed. All neighbours (using 4-connectedness) whose intensity value is within n of the seed's intensity (a value of 10 was used for n) are joined to the seed's region. This continues iteratively outward while the intensity values remain similar. This produces a set of initial regions (ranging in number from 3328 for Ashcroft to 5959 for Cranbrook).

The next phase continually merges regions that have an affinity towards each other. For each region (Ri), a function is applied to it and each of its neighbours (Rj) to indicate their affinity for each other. Affinity is inversely proportional to the value returned by the function. The function used is

$$|I(Ri) - I(Rj)| * (A(Ri) + A(Rj))$$

where I is the average intensity of a region and A is its area. The second term favours the merging of small regions but the similarity of intensities is the major consideration.

Each region (Ra) forms a link with the neighbour that it has the most affinity with (Rb). Now, when Rb calculates affinity values for its neighbours, it may or may not point to Ra. Only those pairs of neighbours that point to each other are candidates for merging. In another departure from Freuder's method, not all doubly linked regions are merged, only the ones with the lowest affinity values (usually the best 30 were merged). This was done because it was found that some of the resulting regions from "high affinity" merges were better candidates for subsequent merges and would have been bypassed by merging some of the "lower affinity" pairs.

This process is repeated for the new regions. Freuder continues iterating until there is only one region left and uses the merge history tree to select regions for subsequent analysis. Processing may stop on its own, however, if there are no doubly linked neighbours to merge (this happened with the Houston image). By displaying the resulting regions at each stage superimposed over the original image, an "iterate until satisfied" termination condition was used in this implementation.

The output from region merging is similar to that from edge detection. A region image is produced where each pixel contains the number identifying the region it belongs to. Also, for each region, a table contains its location, area, perimeter, average intensity, and neighbouring regions.

# APPENDIX D

## Using Mapsee2

The sketch is drawn over an image of the scene using a track ball. The graphic overlay is then translated into a series of drawing commands (plots and gotos), the input form required by Mapsee2 (the algorithm to do this was modified from Stoch, 1981).

Mapsee2 (Havens and Mackworth, 1980; Mackworth and Havens, 1981) reencodes the curves as chains and segments the image. It then finds interpretations for the chains and builds a decomposition hierarchy of the instances.

The instances are Maya schemata which are made up of attribute/value pairs. Since MAIDS is upwards compatible with Maya, the attribute values can be accessed in MISSEE using sgeta, the generic "get attribute" function. The relevant schemata are then written to a file so that they can be transferred from Multilisp to Franz Lisp. The schemata of interest are points, links, lines, chains, bridges, mountains, roads, rivers, towns, road-systems, river-systems, mtn-ranges, and geosystems.

# APPENDIX E

## A Sample Protocol

The following script intermixes user input with the output from MISSEE that constitutes the protocol. All three information sources are used--the intensity image, the sketch map, and the user's input. The resulting instance hierarchy can be found in Figure 6.21.

```
% missys
-> cycle

     ** 1. ***************************************

SUPERVISOR; Input the starting value for the Queue or nil
    e.g. ((100 td %road (smitem *road-3)))

nil          ; use the default

OBJECT: geosystem-top-down. type: nil val: nil

SCHEDULER: adding to Queue (100 river-system-td nil)

     ** 2. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((100 river-system-td nil))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: river-system-top-down. type: nil val: nil

SCHEDULER: adding to Queue (100 bridge-bu-sm (*bridge-1))

     ** 3. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((100 bridge-bu-sm (*bridge-1)))
```

```
  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: bridge-bu-with-sketch-map for *bridge-1
bridge-bu-sm: create a new bridge instance %bridge-1
bridge-bu-sm: *** model consistency for bridge from sm
bridge-bu-sm: regions list =  (130 1186 1629 9 1750 2018)
bridge-bu-sm: order list   =  (O S B B O)
%bridge: value added to smitem =  *bridge-1
bridge-bu-sm: *** model consistency for bridge
bridge-bu-sm: edges list =  (88 11 205 206 241 242)
bridge-bu-sm: edges in proper orientation (88 11 242)
bridge-bu-sm: new edge list =
              ((88 89) (10 11 12 13 14 15) (242 243))
bridge-bu-sm: toomany =  0
bridge-bu-sm: create new (s)curb instance %curb-1
bridge-bu-sm: edge segments are (88 89)
bridge-bu-sm: angles are (248 214)
bridge-bu-sm: length = 20 maxstrength = 5 avgstrength = 3.5
bridge-bu-sm: create new (s)curb instance %curb-2
bridge-bu-sm: edge segments are (10 11 12 13 14 15)
bridge-bu-sm: angles are (90 57 79 45 90 27)
bridge-bu-sm: length = 54 maxstrength = 134 avgstrength = 96.3
bridge-bu-sm: create new (s)curb instance %curb-3
bridge-bu-sm: edge segments are (242 243)
bridge-bu-sm: angles are (228 270)
bridge-bu-sm: length = 20 maxstrength = 22 avgstrength = 18.0
SCHEDULER: adding to Queue (68 river-td (2018 region))
SCHEDULER: adding to Queue (68 river-td (1750 region))
SCHEDULER: adding to Queue (68 river-td (9 region))
SCHEDULER: adding to Queue (68 river-td (130 region))
bridge-bu-sm: neighbouring river regions are nil
SCHEDULER: adding to Queue (68 road-td (1629 region))
bridge-bu-sm: neighbouring road regions are nil
bridge-bu-sm: %bridge-1 is model-consistent
bridge-bu-sm: %bridge-1 must-be-part of some road-system
              and river-system
SCHEDULER: adding to Queue (73 road-system-bu (%bridge-1))
SCHEDULER: adding to Queue (73 river-system-bu (%bridge-1))

     ** 4. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 road-system-bu (%bridge-1))
          (73 river-system-bu (%bridge-1))
          (68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n
```

```
OBJECT: %road-system bottom-up for %bridge-1
%road-system-bu: existing systems  nil
%road-system-bu: componentlist =
                 (%curb-1 %curb-2 %curb-3 %bridge-1)
%road-system-bu: new schema created %road-system-1
%road-system-bu: component added %bridge-1
SCHEDULER: adding to Queue (73 landmass-bu (%road-system-1))

     ** 5. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 river-system-bu (%bridge-1))
          (73 landmass-bu (%road-system-1))
          (68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %river-system bottom-up for %bridge-1
%river-system-bu: existing systems  nil
%river-system-bu: componentlist =
                  (%curb-1 %curb-2 %curb-3 %bridge-1)
%river-system-bu: new schema created %river-system-1
%river-system-bu: component added %bridge-1
SCHEDULER: adding to Queue (73 waterbody-bu (%river-system-1))

     ** 6. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 landmass-bu (%road-system-1))
          (73 waterbody-bu (%river-system-1))
          (68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %landmass bottom-up for %road-system-1
%landmass-bu: existing systems  nil
%landmass-bu: componentlist =
              (%curb-1 %curb-2 %curb-3 %bridge-1 %road-system-1)
%landmass-bu: new schema created %landmass-1
%landmass-bu: component added %road-system-1
SCHEDULER: adding to Queue (73 geosystem-bu (%landmass-1))

     ** 7. ****************************************
```

```
SCHEDULER: Here is the priority queue:

%QUEUE = ((73 waterbody-bu (%river-system-1))
          (73 geosystem-bu (%landmass-1))
          (68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: waterbody-bottom-up for %river-system-1
waterbody-bu: new schema created %waterbody-1
waterbody-bu: adding component %waterbody-1
SCHEDULER: adding to Queue (73 geosystem-bu (%waterbody-1))

     ** 8. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 geosystem-bu (%landmass-1))
          (73 geosystem-bu (%waterbody-1))
          (68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: geosystem-bottom-up for %landmass-1
geosystem-bu: existing systems  nil
geosystem-bu: new schema created %geosystem-1
geosystem-bu: componentlist =
              (%curb-1 %curb-2 %curb-3 %bridge-1
               %road-system-1 %landmass-1 %geosystem-1)

     ** 9. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 geosystem-bu (%waterbody-1))
          (68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: geosystem-bottom-up for %waterbody-1
geosystem-bu: existing systems  %geosystem-1
```

```
geosystem-bu: new schema created %geosystem-2
geosystem-bu: componentlist =
              (%curb-1 %curb-2 %curb-3 %bridge-1
               %river-system-1 %waterbody-1 %geosystem-2)
geosystem-bu: new superior system created: %geosystem-3
              containing %geosystem-2 and %geosystem-1

                ; at this point, the entire structure above
                ; %bridge-1 has been constructed

     ** 10. *************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 river-td (2018 region))
          (68 river-td (1750 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num   --Delete element num
          a s-exp --Add s-exp to the Queue using its priority
          m n1 n2 --Move element n1 to after n2
          q       --Quit processing
          e       --End modifying Queue, continue

1. -> (68 river-td (2018 region))
2. -> (68 river-td (1750 region))
3. -> (68 river-td (9 region))
4. -> (68 river-td (130 region))
5. -> (68 road-td (1629 region))

Command:m 2 0     ; try region 1750 first

SCHEDULER: What is the new priority for 2?72

1. -> (72 river-td (1750 region))
2. -> (68 river-td (2018 region))
3. -> (68 river-td (9 region))
4. -> (68 river-td (130 region))
5. -> (68 road-td (1629 region))

Command:e

OBJECT: Top Down on %river. type: region val: 1750

SCHEDULER: adding to Queue (72 river-bu-sm (*river-5))
SCHEDULER: adding to Queue (72 river-bu-sm (*river-2))

     ** 11. *************************************
```

```
SCHEDULER: Here is the priority queue:

%QUEUE = ((72 river-bu-sm (*river-5))
          (72 river-bu-sm (*river-2))
          (68 river-td (2018 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?l

SUPERVISOR: Do your lisp thing--type d or q to stop

?%printlevel
50
?(setq %printlevel 11)
11
?d

     ** 12. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((72 river-bu-sm (*river-5))
          (72 river-bu-sm (*river-2))
          (68 river-td (2018 region))
          (68 river-td (9 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (72 river-bu-sm (*river-5))
2. -> (72 river-bu-sm (*river-2))
3. -> (68 river-td (2018 region))
4. -> (68 river-td (9 region))
5. -> (68 river-td (130 region))
6. -> (68 road-td (1629 region))

Command:d 1  ; *river-5 is an unintended interpretation

1. -> nil
2. -> (72 river-bu-sm (*river-2))
3. -> (68 river-td (2018 region))
4. -> (68 river-td (9 region))
5. -> (68 river-td (130 region))
6. -> (68 road-td (1629 region))

Command:d 4

1. -> nil
```

```
2. -> (72 river-bu-sm (*river-2))
3. -> (68 river-td (2018 region))
4. -> nil
5. -> (68 river-td (130 region))
6. -> (68 road-td (1629 region))

Command:e

OBJECT: river-bu-with-sketch-map for *river-2
river-bu-sm: create a new river instance %river-1
river-bu-sm: *** model consistency for river from sm
river-bu-sm: regions list =  (1750)
river-bu-sm: interpretations are (WATER)
river-bu-sm: one interpretation must be consistent with water
             or fail
river-bu-sm: added neighbouring regions are (2295 2095 2018)
%river: value added to regions = (1750 2295 2095 2018)
river-bu-sm: %river-1 is model-consistent
%river: value added to smitem = *river-2
river-bu-sm: neighbouring bridge regions are (%bridge-1)
river-bu-sm: %river-1 must-be-part of some river-system
SCHEDULER: adding to Queue (73 river-system-bu (%river-1))

     ** 13. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 river-system-bu (%river-1))
          (68 river-td (2018 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %river-system bottom-up for %river-1

%river-system-bu: existing systems  %river-system-1
%river-system-bu: %river-1 added to %river-system-1

                ; %river-1 joins the same river-system that
                ; contains %bridge-1

     ** 14. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 river-td (2018 region))
          (68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: Top Down on %river. type: region val: 2018
```

```
SCHEDULER: adding to Queue (68 river-bu (2018))

     ** 15. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 river-td (130 region))
          (68 road-td (1629 region))
          (68 river-bu (2018))))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (68 river-td (130 region))
2. -> (68 road-td (1629 region))
3. -> (68 river-bu (2018))

Command:m 3 0

SCHEDULER: What is the new priority for 3?72

1. -> (72 river-bu (2018))
2. -> (68 river-td (130 region))
3. -> (68 road-td (1629 region))

Command:e

OBJECT: river-bu-image-alone
river-bu-im: region 2018 has already been tried
             with result %river-1

     ** 16. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 river-td (130 region))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: Top Down on %river. type: region val: 130
SCHEDULER: adding to Queue (68 river-bu-sm (*river-5))
SCHEDULER: adding to Queue (68 river-bu-sm (*river-1))

     ** 17. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 road-td (1629 region))
          (68 river-bu-sm (*river-5))
          (68 river-bu-sm (*river-1)))
  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (68 road-td (1629 region))
2. -> (68 river-bu-sm (*river-5))
3. -> (68 river-bu-sm (*river-1))

Command:d 2

1. -> (68 road-td (1629 region))
2. -> nil
3. -> (68 river-bu-sm (*river-1))

Command:m 3 0

SCHEDULER: What is the new priority for 3?75

1. -> (75 river-bu-sm (*river-1))
2. -> (68 road-td (1629 region))

Command:e

OBJECT: river-bu-with-sketch-map for *river-1
river-bu-sm: create a new river instance %river-2
river-bu-sm: *** model consistency for river from sm
river-bu-sm: regions list =  (130 943)
river-bu-sm: interpretations are (WATER WATER)
river-bu-sm: one interpretation must be consistent with water
             or fail
river-bu-sm: added neighbouring regions are
             (2065 1872 1779 1058 874 725 524 208 873)
%river: value added to regions =
             (943 130 2065 1872 1779 1058 874 725 524 208 873)
river-bu-sm: %river-2 is model-consistent
%river: value added to smitem = *river-1
river-bu-sm: neighbouring bridge regions are (%bridge-1)
SCHEDULER: adding to Queue (70 bridge-td (8))
river-bu-sm: %river-2 must-be-part of some river-system
SCHEDULER: adding to Queue (73 river-system-bu (%river-2))

     ** 18. *************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((73 river-system-bu (%river-2))
          (70 bridge-td (8))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %river-system bottom-up for %river-2
%river-system-bu: existing systems  %river-system-1
```

```
%river-system-bu: %river-2 added to %river-system-1

     ** 19. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((70 bridge-td (8))
          (68 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (70 bridge-td (8))
2. -> (68 road-td (1629 region))

Command:d 1

1. -> nil
2. -> (68 road-td (1629 region))

Command:e

OBJECT: Top Down on %road. type: region val: 1629
SCHEDULER: adding to Queue (68 road-bu-sm (*road-9))
SCHEDULER: adding to Queue (68 road-bu-sm (*road-5))

     ** 20. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 road-bu-sm (*road-9))
          (68 road-bu-sm (*road-5)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (68 road-bu-sm (*road-9))
2. -> (68 road-bu-sm (*road-5))

Command:d 1

1. -> nil
2. -> (68 road-bu-sm (*road-5))

Command:e     ; *road-5 is the road that runs over *bridge-1

OBJECT: road-bu-with-sketch-map for *road-5
road-bu-sm: create a new road instance %road-1
road-bu-sm: *** model consistency for new-road from sm
road-bu-sm: deviance = 53.4132
road-bu-sm: create new curb instance %curb-4
```

```
road-bu-sm: edge segments are (253 255) angles (153 140)
road-bu-sm: length = 16 maxstrength = 57 avgstrength = 32.5
road-bu-sm: create new curb instance %curb-5
road-bu-sm: edge segments are (263) angles (136)
road-bu-sm: length = 3 maxstrength = 34 avgstrength = 34.0
road-bu-sm: create new curb instance %curb-6
road-bu-sm: edge segments are (260 261) angles (136 172)
road-bu-sm: length = 15 maxstrength = 3 avgstrength = 2.5
                                  .
                                  .
                                  .
road-bu-sm: create new curb instance %curb-36
road-bu-sm: edge segments are (351) angles (120)
road-bu-sm: length = 15 maxstrength = 42 avgstrength = 42.0
road-bu-sm: create new curb instance %curb-37
road-bu-sm: edge segments are (50) angles (142)
road-bu-sm: length = 10 maxstrength = 47 avgstrength = 47.0
road-bu-sm: regions list =  (2575 2432 1629 9 738 19 18)
road-bu-sm: interpretations are
            ((URBAN HILLS) (URBAN HILLS) (ROAD MOUNTAIN)
             (URBAN HILLS) (URBAN HILLS) (URBAN HILLS)
             (ROAD MOUNTAIN))
road-bu-sm: one interpretation must be consistent with road
            or fail
%road: value added to smitem = *road-5
SCHEDULER: adding to Queue (50 town-td (2575 region))
SCHEDULER: adding to Queue (50 town-td (2432 region))
SCHEDULER: adding to Queue (50 town-td (9 region))
SCHEDULER: adding to Queue (50 town-td (738 region))
SCHEDULER: adding to Queue (50 town-td (19 region))
%road: value added to regions = (18 1629)
road-bu-sm: %road-1 is model-consistent
road-bu-sm: neighbouring bridge regions are (%bridge-1)
road-bu-sm: neighbouring road regions are nil
SCHEDULER: adding to Queue (50 road-td (18 region))
SCHEDULER: adding to Queue (50 road-td (1629 region))
road-bu-sm: neighbouring town regions are nil
road-bu-sm: %road-1 must-be-part of some road-system
SCHEDULER: adding to Queue (53 road-system-bu (%road-1))

     ** 21. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((53 road-system-bu (%road-1))
          (50 town-td (2575 region))
          (50 town-td (2432 region))
          (50 town-td (9 region))
          (50 town-td (738 region))
          (50 town-td (19 region))
          (50 road-td (18 region))
          (50 road-td (1629 region)))
```

```
  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %road-system bottom-up for %road-1
%road-system-bu: existing systems  %road-system-1
%road-system-bu: %road-1 added to %road-system-1

     ** 22. ************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((50 town-td (2575 region))
          (50 town-td (2432 region))
          (50 town-td (9 region))
          (50 town-td (738 region))
          (50 town-td (19 region))
          (50 road-td (18 region))
          (50 road-td (1629 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (50 town-td (2575 region))
2. -> (50 town-td (2432 region))
3. -> (50 town-td (9 region))
4. -> (50 town-td (738 region))
5. -> (50 town-td (19 region))
6. -> (50 road-td (18 region))
7. -> (50 road-td (1629 region))

Command:d 1

1. -> nil
2. -> (50 town-td (2432 region))
3. -> (50 town-td (9 region))
4. -> (50 town-td (738 region))
5. -> (50 town-td (19 region))
6. -> (50 road-td (18 region))
7. -> (50 road-td (1629 region))

Command:d 2

1. -> nil
2. -> nil
3. -> (50 town-td (9 region))
4. -> (50 town-td (738 region))
5. -> (50 town-td (19 region))
6. -> (50 road-td (18 region))
7. -> (50 road-td (1629 region))

Command:e

OBJECT: Top Down on %town. type: region val: 9
```

```
SCHEDULER: adding to Queue (50 town-bu-sm (*town-1))

     ** 23. *************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((50 town-td (738 region))
          (50 town-td (19 region))
          (50 road-td (18 region))
          (50 road-td (1629 region))
          (50 town-bu-sm (*town-1)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. —> (50 town-td (738 region))
2. —> (50 town-td (19 region))
3. —> (50 road-td (18 region))
4. —> (50 road-td (1629 region))
5. —> (50 town-bu-sm (*town-1))

Command:m 5 0

SCHEDULER: What is the new priority for 5?70

1. —> (70 town-bu-sm (*town-1))
2. —> (50 town-td (738 region))
3. —> (50 town-td (19 region))
4. —> (50 road-td (18 region))
5. —> (50 road-td (1629 region))

Command:e

OBJECT: town-bu-with-sketch-map for *town-1
town-bu-sm: create a new town instance %town-1
town-bu-sm: *** model consistency for town from sm
town-bu-sm: regions list =  (9)
town-bu-sm: interpretations are ((URBAN HILLS))
town-bu-sm: one interpretation must be consistent with urban
            or fail
town-bu-sm: %town-1 is model-consistent
%town: value added to regions = (9)
town-bu-sm: centre of town at (51 . 81)
%town: value added to smitem = *town-1
town-bu-sm: neighbouring road regions are nil
SCHEDULER: adding to Queue (50 road-td (8))
SCHEDULER: adding to Queue (50 road-td (1006))
town-bu-sm: %town-1 must-be-part of some landmass
SCHEDULER: adding to Queue (53 landmass-bu (%town-1))
DEMON: adding neighbours link from %road-1 to %town-1

     ** 24. *************************************
```

```
SCHEDULER: Here is the priority queue:

%QUEUE = ((53 landmass-bu (%town-1))
          (50 town-td (738 region))
          (50 town-td (19 region))
          (50 road-td (18 region))
          (50 road-td (1629 region))
          (50 road-td (8))
          (50 road-td (1006)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?l

SUPERVISOR: Do your lisp thing--type d or q to stop

?(sprint '%town-1)

****instance: %town-1   ***of stereotype: %town
              -------

centre:      value: (51 . 81)     %confidence:  100

regions:     value: (9)           %confidence:  100

             %if-added:    (prog nil
                              (printlb 8 "%town: value
                               added to" "regions =" %val))

smitem:      value: *town-1       %confidence:  100

             %if-added:    (prog nil
                              (printlb 8 "%town: value
                               added to" "smitem =" %val))

apo->              nil
decomposes-to->    nil
neighbours->       %road-1
conf-alg:    (prog nil
                (return (cond ((sgetv %name 'smitem 'n) 50)
                              (t 25))))
confidence: 50
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nil
?d

     ** 25. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((53 landmass-bu (%town-1))
          (50 town-td (738 region))
          (50 town-td (19 region))
          (50 road-td (18 region))
          (50 road-td (1629 region))
```

```
                 (50 road-td (8))
                 (50 road-td (1006)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (53 landmass-bu (%town-1))
2. -> (50 town-td (738 region))
3. -> (50 town-td (19 region))
4. -> (50 road-td (18 region))
5. -> (50 road-td (1629 region))
6. -> (50 road-td (8))
7. -> (50 road-td (1006))

Command:d 7

1. -> (53 landmass-bu (%town-1))
2. -> (50 town-td (738 region))
3. -> (50 town-td (19 region))
4. -> (50 road-td (18 region))
5. -> (50 road-td (1629 region))
6. -> (50 road-td (8))
7. -> nil

Command:d 6

1. -> (53 landmass-bu (%town-1))
2. -> (50 town-td (738 region))
3. -> (50 town-td (19 region))
4. -> (50 road-td (18 region))
5. -> (50 road-td (1629 region))
6. -> nil
7. -> nil

Command:d 5

1. -> (53 landmass-bu (%town-1))
2. -> (50 town-td (738 region))
3. -> (50 town-td (19 region))
4. -> (50 road-td (18 region))
5. -> nil
6. -> nil
7. -> nil

Command:m 4 0

SCHEDULER: What is the new priority for 4?60

1. -> (60 road-td (18 region))
2. -> (53 landmass-bu (%town-1))
3. -> (50 town-td (738 region))
4. -> (50 town-td (19 region))
```

```
Command:e     ; region 18 contains several roads

OBJECT: Top Down on %road. type: region val: 18
SCHEDULER: adding to Queue (60 road-bu-sm (*road-3))
SCHEDULER: adding to Queue (60 road-bu-sm (*road-4))
SCHEDULER: adding to Queue (60 road-bu-sm (*road-8))
SCHEDULER: adding to Queue (60 road-bu-sm (*road-7))
SCHEDULER: adding to Queue (60 road-bu-sm (*road-6))
SCHEDULER: adding to Queue (60 road-bu-sm (*road-5))

     ** 26. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((60 road-bu-sm (*road-3))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (738 region))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (60 road-bu-sm (*road-3))
2. -> (60 road-bu-sm (*road-4))
3. -> (60 road-bu-sm (*road-8))
4. -> (60 road-bu-sm (*road-7))
5. -> (60 road-bu-sm (*road-6))
6. -> (60 road-bu-sm (*road-5))
7. -> (53 landmass-bu (%town-1))
8. -> (50 town-td (738 region))
9. -> (50 town-td (19 region))

Command:m 8 0

SCHEDULER: What is the new priority for 8?70

1. -> (70 town-td (738 region))
2. -> (60 road-bu-sm (*road-3))
3. -> (60 road-bu-sm (*road-4))
4. -> (60 road-bu-sm (*road-8))
5. -> (60 road-bu-sm (*road-7))
6. -> (60 road-bu-sm (*road-6))
7. -> (60 road-bu-sm (*road-5))
8. -> (53 landmass-bu (%town-1))
9. -> (50 town-td (19 region))

Command:e
```

Appendix E

```
OBJECT: Top Down on %town. type: region val: 738
SCHEDULER: adding to Queue (70 town-bu-im (738))

     ** 27. *************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((70 town-bu-im (738))
          (60 road-bu-sm (*road-3))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: town-bu-image-alone
town-bu-im: region =  738 area 2500
town-bu-im: create a new town instance %town-2
town-bu-im: *** model consistency for town from sm
%town: value added to regions = 738
town-bu-im: centre of town at (33 . 66)
town-bu-im: %town-2 is model-consistent
town-bu-im: %town-2 must-be-part of some landmass
SCHEDULER: adding to Queue (68 landmass-bu (%town-2))
town-bu-im: neighbouring road regions are nil
DEMON: adding neighbours link from %road-1 to %town-2

     ** 28. *************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 landmass-bu (%town-2))
          (60 road-bu-sm (*road-3))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (68 landmass-bu (%town-2))
2. -> (60 road-bu-sm (*road-3))
3. -> (60 road-bu-sm (*road-4))
4. -> (60 road-bu-sm (*road-8))
```

```
5. -> (60 road-bu-sm (*road-7))
6. -> (60 road-bu-sm (*road-6))
7. -> (60 road-bu-sm (*road-5))
8. -> (53 landmass-bu (%town-1))
9. -> (50 town-td (19 region))

Command:m 2 0

SCHEDULER: What is the new priority for 2?75

1. -> (75 road-bu-sm (*road-3))
2. -> (68 landmass-bu (%town-2))
3. -> (60 road-bu-sm (*road-4))
4. -> (60 road-bu-sm (*road-8))
5. -> (60 road-bu-sm (*road-7))
6. -> (60 road-bu-sm (*road-6))
7. -> (60 road-bu-sm (*road-5))
8. -> (53 landmass-bu (%town-1))
9. -> (50 town-td (19 region))

Command:e     ; *road-3 runs "north" from *town-1

OBJECT: road-bu-with-sketch-map for *road-3
road-bu-sm: create a new road instance %road-2
road-bu-sm: *** model consistency for road from sm
road-bu-sm: deviance = 0.632455
road-bu-sm: create new curb instance %curb-38
road-bu-sm: edge segments are (412) angles (104)
road-bu-sm: length = 6 maxstrength = 73 avgstrength = 73.0
road-bu-sm: one orientation must be consistent with other
            curbs or fail
road-bu-sm: create new curb instance %curb-39
road-bu-sm: edge segments are (408) angles (90)
road-bu-sm: length = 3 maxstrength = 43 avgstrength = 43.0
road-bu-sm: max confidence is 50.0
road-bu-sm: regions list =  (18)
road-bu-sm: interpretations are ((ROAD MOUNTAIN))
road-bu-sm: one interpretation must be consistent with road
            or fail
%road: value added to smitem = *road-3
SYSTEM: Warning, more than one interpretation for region 18
SYSTEM:   Adding %road %road-2 to (%road %road-1)
%road: value added to regions = (18)
road-bu-sm: %road-2 is model-consistent
road-bu-sm: neighbouring bridge regions are nil
road-bu-sm: neighbouring road regions are nil
SCHEDULER: adding to Queue (67 road-td (18 region))
road-bu-sm: neighbouring town regions are (%town-1)
road-bu-sm: %road-2 must-be-part of some road-system
SCHEDULER: adding to Queue (70 road-system-bu (%road-2))

     ** 29. ************************************
```

```
SCHEDULER: Here is the priority queue:

%QUEUE = ((70 road-system-bu (%road-2))
          (68 landmass-bu (%town-2))
          (67 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (70 road-system-bu (%road-2))
2. -> (68 landmass-bu (%town-2))
3. -> (67 road-td (18 region))
4. -> (60 road-bu-sm (*road-4))
5. -> (60 road-bu-sm (*road-8))
6. -> (60 road-bu-sm (*road-7))
7. -> (60 road-bu-sm (*road-6))
8. -> (60 road-bu-sm (*road-5))
9. -> (53 landmass-bu (%town-1))
10. -> (50 town-td (19 region))

Command:a (77 road-bu-sm (*road-4))

1. -> (77 road-bu-sm (*road-4))
2. -> (70 road-system-bu (%road-2))
3. -> (68 landmass-bu (%town-2))
4. -> (67 road-td (18 region))
5. -> (60 road-bu-sm (*road-4))
6. -> (60 road-bu-sm (*road-8))
7. -> (60 road-bu-sm (*road-7))
8. -> (60 road-bu-sm (*road-6))
9. -> (60 road-bu-sm (*road-5))
10. -> (53 landmass-bu (%town-1))
11. -> (50 town-td (19 region))

Command:e      ; *road-4 starts at *road-5 and runs by
               ; the end of *road-2

OBJECT: road-bu-with-sketch-map for *road-4
road-bu-sm: create a new road instance %road-3
road-bu-sm: *** model consistency for road from sm
road-bu-sm: deviance = 0.921442
road-bu-sm: create new curb instance %curb-40
road-bu-sm: edge segments are (422) angle (24)
road-bu-sm: length = 15 maxstrength = 74 avgstrength = 74.0
road-bu-sm: one orientation must be consistent with other
```

```
            curbs or fail
road-bu-sm: create new curb instance %curb-41
road-bu-sm: edge segments are (411) angles (162)
road-bu-sm: length = 5 maxstrength = 41 avgstrength = 41.0
                           .
                           .
                           .
road-bu-sm: create new curb instance %curb-50
road-bu-sm: edge segments are (146) angles (39)
road-bu-sm: length = 18 maxstrength = 14 avgstrength = 14.0
road-bu-sm: create new curb instance %curb-51
road-bu-sm: edge segments are (160 159 161)
road-bu-sm: curbangles are (45 180 180)
road-bu-sm: length = 12 maxstrength = 86 avgstrength = 74.667
road-bu-sm: max confidence is 53.986
road-bu-sm: regions list =  (438 738 18)
road-bu-sm: interpretations are ((URBAN HILLS) (URBAN HILLS)
            (ROAD MOUNTAIN))
road-bu-sm: one interpretation must be consistent with road
            or fail
%road: value added to smitem = *road-4
SCHEDULER: adding to Queue (64 town-td (438 region))
SYSTEM: Warning, more than one interpretation for region 18
SYSTEM: Adding %road %road-3 to (%road %road-2 %road %road-1)
%road: value added to regions = (18)
road-bu-sm: %road-3 is model-consistent
road-bu-sm: neighbouring bridge regions are nil
road-bu-sm: neighbouring road regions are (%road-1 %road-2)
SCHEDULER: adding to Queue (64 road-td (18 region))
road-bu-sm: neighbouring town regions are nil
road-bu-sm: %road-3 must-be-part of some road-system
SCHEDULER: adding to Queue (67 road-system-bu (%road-3))

     ** 30. ***************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((70 road-system-bu (%road-2))
          (68 landmass-bu (%town-2))
          (67 road-system-bu (%road-3))
          (67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?l
```

```
SUPERVISOR: Do your lisp thing--type d or q to stop

?(sprint '%road-3)

****instance: %road-3   ***of stereotype: %road
              -------

smitem:              value: *road-4      %confidence: 100
                     %if-added:          (prog ...

ends:                value: ((41 . 62) (29 . 126))
                     %confidence:          100

separation:          value: nil          %confidence: nil

deviance:            value: 0.921442     %confidence: nil

regions:             value: (18)         %confidence: 100
                     %if-added:          (prog ...

apo->            nil
decomposes-to->  (%curb-40 %curb-41 %curb-42 %curb-43 %curb-44
                  %curb-45 %curb-46 %curb-47 %curb-48 %curb-48
                  %curb-49 %curb-50 %curb-51)
neighbours->     (%road-1 %road-2)
conf-alg:           (prog (val lst) ...
confidence:         64
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nil
?d

     ** 31. *************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((70 road-system-bu (%road-2))
          (68 landmass-bu (%town-2))
          (67 road-system-bu (%road-3))
          (67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %road-system bottom-up for %road-2
%road-system-bu: existing systems  %road-system-1
```

```
%road-system-bu: new schema created %road-system-2
%road-system-bu: component added %road-2
SCHEDULER: adding to Queue (68_landmass-bu (%road-system-2))

     ** 32. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((68 landmass-bu (%road-system-2))
          (68 landmass-bu (%town-2))
          (67 road-system-bu (%road-3))
          (67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (68 landmass-bu (%road-system-2))
2. -> (68 landmass-bu (%town-2))
3. -> (67 road-system-bu (%road-3))
4. -> (67 road-td (18 region))
5. -> (64 town-td (438 region))
6. -> (64 road-td (18 region))
7. -> (60 road-bu-sm (*road-8))
8. -> (60 road-bu-sm (*road-7))
9. -> (60 road-bu-sm (*road-6))
10. -> (60 road-bu-sm (*road-5))
11. -> (53 landmass-bu (%town-1))
12. -> (50 town-td (19 region))))

Command:d 1

1. -> nil
2. -> (68 landmass-bu (%town-2))
3. -> (67 road-system-bu (%road-3))
4. -> (67 road-td (18 region))
5. -> (64 town-td (438 region))
6. -> (64 road-td (18 region))
7. -> (60 road-bu-sm (*road-8))
8. -> (60 road-bu-sm (*road-7))
9. -> (60 road-bu-sm (*road-6))
10. -> (60 road-bu-sm (*road-5))
11. -> (53 landmass-bu (%town-1))
12. -> (50 town-td (19 region))))
```

```
Command:e

OBJECT: %landmass bottom-up for %town-2
%landmass-bu: existing systems  (%landmass-1)
%landmass-bu: %town-2 added to %landmass-1

     ** 33. **************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((67 road-system-bu (%road-3))
          (67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (53 landmass-bu (%town-1))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?y

Commands: d num, a s-exp, m n1 n2, q, or e

1. -> (67 road-system-bu (%road-3))
2. -> (67 road-td (18 region))
3. -> (64 town-td (438 region))
4. -> (64 road-td (18 region))
5. -> (60 road-bu-sm (*road-4))
6. -> (60 road-bu-sm (*road-8))
7. -> (60 road-bu-sm (*road-7))
8. -> (60 road-bu-sm (*road-6))
9. -> (60 road-bu-sm (*road-5))
10. -> (53 landmass-bu (%town-1))
11. -> (50 town-td (19 region))

Command:m 10 1

SCHEDULER: What is the new priority for 10?67

1. -> (67 road-system-bu (%road-3))
2. -> (67 landmass-bu (%town-1))
3. -> (67 road-td (18 region))
4. -> (64 town-td (438 region))
5. -> (64 road-td (18 region))
6. -> (60 road-bu-sm (*road-4))
7. -> (60 road-bu-sm (*road-8))
8. -> (60 road-bu-sm (*road-7))
9. -> (60 road-bu-sm (*road-6))
10. -> (60 road-bu-sm (*road-5))
11. -> (50 town-td (19 region))
```

```
Command:e

OBJECT: %road-system bottom-up for %road-3
%road-system-bu: existing systems  (%road-system-2
                 %road-system-1)
%road-system-bu: %road-3 added to %road-system-2
%road-system-bu: merging %road-system-1 into %road-system-2

     ** 34. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((67 landmass-bu (%town-1))
          (67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?n

OBJECT: %landmass bottom-up for %town-1
%landmass-bu: existing systems  (%landmass-1)
%landmass-bu: %town-1 added to %landmass-1

     ** 35. ****************************************

SCHEDULER: Here is the priority queue:

%QUEUE = ((67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (50 town-td (19 region)))

  Do you want to modify it(y), be in lisp(l), or quit(q)?l

?(sprint '%bridge-1)
****instance: %bridge-1   ***of stereotype: %bridge
              ---------

smitem:              value: *bridge-1    %confidence: 100
                     %if-added:         (prog ...

orderlist:           value: (O S B B O)  %confidence: 100
```

```
neighbourregions:   value: (2018 1750 9 130)
                    %confidence:        100

shadowregions:      value: (1186)       %confidence: 100

roadregions:        value: (1629)       %confidence: 100

decomposes-to->   (%curb-3 %curb-2 %curb-1)
neighbours->      (%road-1 %river-2 %river-1)
apo->             (%road-system-2 %river-system-1)
conf-alg:           (prog ...
confidence:         68
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nil
?(sprint '%curb-1)
****instance: %curb-1   ***of stereotype: %curb
              -------

avgstrength:        value: 3.5         %confidence: 75

maxstrength:        value: 5           %confidence: 75

length:             value: 20          %confidence: 100

angles:             value: (248 214)   %confidence: (80)

edgesegs:           value: (88 89)     %confidence: 100

type:               value: shadow      %confidence: 100

apo->             %bridge-1
conf-alg:           (prog ...
confidence:         85
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nil
?(sprint '%curb-2)
****instance: %curb-2   ***of stereotype: %curb
              -------

avgstrength:        value: 96.33       %confidence: 100

maxstrength:        value: 134         %confidence: 100

length:             value: 54          %confidence: 100

angles:             value: (90 57 79 45 90 27)
                    %confidence:        (80)

edgesegs:           value: (10 11 12 13 14 15)
                    %confidence:        100

type:               value: road        %confidence: 100
```

```
apo->           %bridge-1
conf-alg:           (prog ...
confidence:         93
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nil
?(sprint '%curb-3)
****instance: %curb-3   ***of stereotype: %curb
              -------

avgstrength:        value: 18.0       %confidence: 100

maxstrength:        value: 22         %confidence: 100

length:             value: 20         %confidence: 100

angles:             value: (228 270)  %confidence: (80)

edgesegs:           value: (242 243)  %confidence: 100

type:               value: road       %confidence: 100

apo->           %bridge-1
conf-alg:           (prog ...
confidence:         93
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

nil
?%instances
(%landmass-1 %road-system-2 %curb-9 %curb-8 %road-3 %curb-7
 %curb-6 %road-2 %town-2 %town-1 %curb-5 %curb-4 %road-1
 %river-2 %river-1 %geosystem-3 %geosystem-2 %geosystem-1
 %waterbody-1 %river-system-1 %curb-3 %curb-2 %curb-1
 %bridge-1)
?(sprint '%road-system-2)
****instance: %road-system-2   ***of stereotype: %road-system
              --------------

smitem:             value: nil        %confidence: nil
                    %if-added:          (prog ...

neighbours->    nil
decomposes-to-> (%bridge-1 %road-1 %road-3 %road-2)
apo->           %landmass-1
conf-alg:           (prog ...
confidence:         63
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

?(sprint '%river-1)
****instance: %river-1   ***of stereotype: %river
              --------

regions:            value: (1750 2295 2095 2018)
                    %confidence:        100
                    %if-added:         (prog ...
```

```
smitem:                 value: *river-2   %confidence: 100
                        %if-added:         (prog ...

apo->              %river-system-1
decomposes-to->    nil
neighbours->       %bridge-1
conf-alg:             (prog ...
confidence:           70
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
?(sprint '%landmass-1)
****instance: %landmass-1   ***of stereotype: %landmass
              -----------

smitem:                 value: nil        %confidence: nil
                        %if-added:         (prog ...

neighbours->       nil
decomposes-to->    (%town-1 %road-system-2)
ako->              %geosystem-1
conf-alg:             (prog ...
confidence:           63
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
?(ssprintn '%geosystem-3)
%geosystem-3
---aio---
%geosystem
  ---instances---
  %geosystem-3
  %geosystem-2
    ---apo---
    %geosystem-3
    ---specializes-to---
    %waterbody-1
      ---ako---
      %geosystem-2
      ---aio---
      %waterbody
        ---instances---
        %waterbody-1
      ---specializes-to---
      %river-system-1
        ---aio---
        %river-system
          ---instances---
          %river-system-1
        ---ako---
        %waterbody-1
        ---specializes-to---
        %river-2
          ---apo---
          %river-system-1
          ---aio---
          %river
```

```
---instances---
%river-2
%river-1
  ---apo---
  %river-system-1
  ---aio---
  %river
  ---neighbours---
  %bridge-1
    ---aio---
    %bridge
      ---instances---
      %bridge-1
    ---decomposes-to---
    %curb-3
      ---apo---
      %bridge-1
      ---aio---
      %curb
        ---instances---
        %curb-51
          ---apo---
          %road-3
            ---apo---
            %road-system-2
              ---apo---
              %landmass-1
                ---ako---
                %geosystem-1
                  ---apo---
                  %geosystem-3
                  ---specializes-to---
                  %landmass-1
                  ---aio---
                  %geosystem
                ---aio---
                %landmass
                  ---instances---
                  %landmass-1
                ---decomposes-to---
                %road-system-2
                %town-2
                  ---apo---
                  %landmass-1
                  ---aio---
                  %town
                    ---instances---
                    %town-2
                    %town-1
                       ---apo---
                       %landmass-1
                       ---aio---
                       %town
```

```
         ---neighbours---
         %road-2
           ---apo---
           %road-system-2
           ---aio---
           %road
             ---instances---
             %road-3
             %road-2
             %road-1
             ---aio---
               %road
               ---decomposes-to---
               %curb-37
               ---apo---
               %road-1
               ---aio---
               %curb
                 .
                 .
                 .
               %curb-4
               ---apo---
               %road-1
               ---aio---
               %curb
               ---neighbours---
               %road-3
               %town-2
               %town-1
               %bridge-1
                 ---apo---
                 %road-system-2
           ---decomposes-to---
           %curb-39
           ---apo---
           %road-2
           ---aio---
           %curb
           %curb-38
           ---apo---
           %road-2
           ---aio---
           %curb
           ---neighbours---
           %road-3
           %town-1
           %road-1
    ---neighbours---
    %road-1
    %town-1
---aio---
%road-system
```

```
                    ---instances---
                    %road-system-2
                  ---decomposes-to---
                  %bridge-1
                  %road-1
                  %road-3
                  %road-2
                ---aio---
                %road
                ---decomposes-to---
                %curb-51
                %curb-50
                  ---apo---
                  %road-3
                  ---aio---
                  %curb
                    .
                    .
                    .
                %curb-40
                  ---apo---
                  %road-3
                  ---aio---
                  %curb
                ---neighbours---
                %road-2
                %road-1
              ---aio---
              %curb
            %curb-50
              .
              .
              .
            %curb-2
              ---apo---
              %bridge-1
              ---aio---
              %curb
            %curb-1
              ---apo---
              %bridge-1
              ---aio---
              %curb
        %curb-2
        %curb-1
        ---neighbours---
        %road-1
        %river-2
        %river-1
        ---apo---
        %road-system-2
        %river-system-1
  ---neighbours---
```

```
          %bridge-1
        %river-1
        %bridge-1
    ---aio---
    %geosystem
  %geosystem-1
---decomposes-to---
%geosystem-1
%geosystem-2
nil
?(sprintn '%river-system-1 'decomposes-to)
%river-system-1
     %river-1
     %river-2
     %bridge-1
          %curb-1
          %curb-2
          %curb-3
nil
?(sprintn '%landmass-1 'decomposes-to)
%landmass-1
     %town-1
     %road-system-2
          %bridge-1
               %curb-1
               %curb-2
               %curb-3
          %road-1
               %curb-4
                  .
                  .
                  .
               %curb-37
          %road-2
               %curb-38
               %curb-39
          %road-3
               %curb-40
                  .
                  .
                  .
               %curb-51
nil
?(sprintn '%geosystem-3 'decomposes-to)
%geosystem-3
     %geosystem-1
     %geosystem-2
nil
?d

     ** 36. **************************************

SCHEDULER: Here is the priority queue:
```

```
%QUEUE = ((67 road-td (18 region))
          (64 town-td (438 region))
          (64 road-td (18 region))
          (60 road-bu-sm (*road-4))
          (60 road-bu-sm (*road-8))
          (60 road-bu-sm (*road-7))
          (60 road-bu-sm (*road-6))
          (60 road-bu-sm (*road-5))
          (50 town-td (19 region)))
  Do you want to modify it(y), be in lisp(l), or quit(q)?q
SUPERVISOR: Cycle broken by user--bye
nil
-> %
```