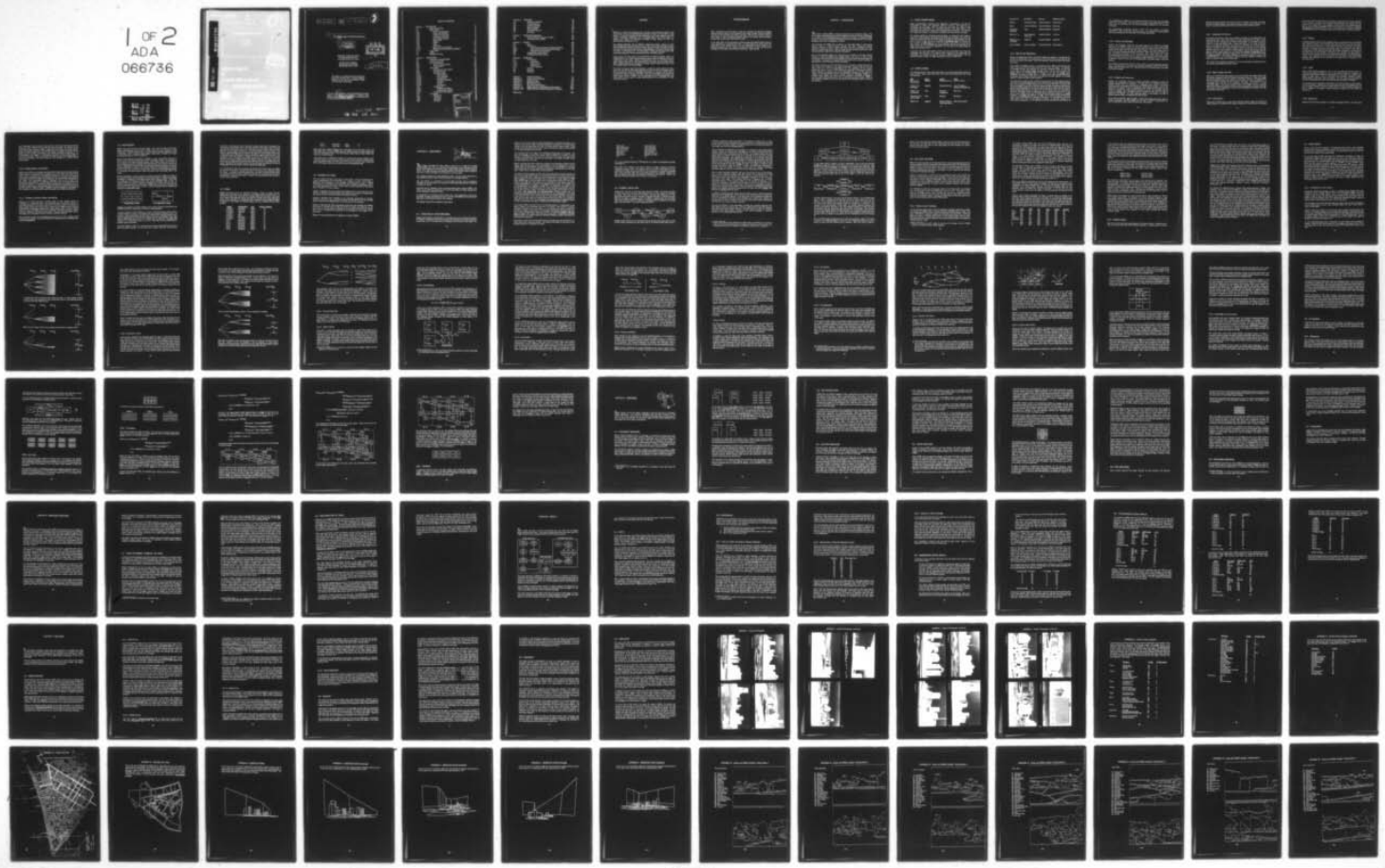
AD-A066 736 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/1
THE ARGOS IMAGE UNDERSTANDING SYSTEM,(U)
NOV 78 S RUBIN F44620-73-C-0074

UNCLASSIFIED AFOSR-TR-79-0087 NL

1 OF 2
AD A
066736



DDC FILE COPY

AD A066736

(18) AFOSR (19) TR-79-0087

(2)

(6) The ARGOS Image Understanding System,

(10) Steven Rubin

(11) Nov[illegible]978

DDC
RECEIVED
APR 2 1979
C

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

403 081

Author's Current Address:
Bell Telephone Laboratories
Holmdel, New Jersey 07733

(12) 149p.

This report is an updated version of the doctoral thesis with the same name. It includes additional research on the use of pre-segmented images and hierarchies of knowledge.

(15) This work was supported by the Advanced Research Projects Agency of the Office of the Secretary of Defense under contract number F44620-73-C-0074 and is monitored by the Air Force Office of Scientific Research.

403 081

79 03 30 059

TABLE OF CONTENTS

Chapter 1	INTRODUCTION	5
1.1	Image Understanding	6
1.2	Other Systems	6
1.2.1	Barrow and Popplestone	7
1.2.2	Fischler and Elschlager	8
1.2.3	Feldman and Yakimovsky	8
1.2.4	Tenenbaum and Barrow	9
1.2.5	Sakai, Kanade, and Ohta	9
1.2.6	Keng and Fu	9
1.2.7	Perkins	10
1.2.8	Waltz	10
1.2.9	Mackworth	10
1.2.10	Ballard, Brown, and Feldman	11
1.2.11	Williams, Lowrance, Hanson, and Riseman	11
1.3	Locus Search	12
1.4	ARGOS	13
1.5	Reading This Thesis	14
Chapter 2	LOCUS SEARCH	15
2.1	Structure of Locus Knowledge	15
2.2	Symbolic use of PPEs	17
2.3	Low Level Matching	20
2.3.1	*Feature Vector Templates*	20
2.3.2	Distance Metrics	22
2.4	Locus Search	24
2.4.1	Introduction to Locus search	24
2.4.2	Overview of Locus	26
2.4.3	The Forward Pass	28
2.4.3.1	Order of Search	28
2.4.3.2	Path Likelihoods	29
2.4.3.3	Recombination	30
2.4.3.4	Transition Likelihoods	31
2.4.3.5	The Beam	32
2.4.3.6	Pruning	32
2.4.3.7	Normalization	33
2.4.4	The Backtrace	33
2.4.4.1	Conflicts The Problem	34
2.4.4.2	Conflicts Some Answers	35
2.4.5	Advantages of Locus Search	37
2.5	An Example	38
2.5.1	Background	38
2.5.2	Low Level	39
2.5.3	The Search	40
2.5.4	Comments	43

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. SPECIAL
A

Chapter 3	KNOWLEDGE	45
3.1	Adjacency Knowledge	45
3.2	Pre-Segmentation	47
3.3	Location Knowledge	47
3.4	Shape Knowledge	48
3.5	Size Knowledge	50
3.6	Knowledge Constraint	51
3.7	Conclusion	52
Chapter 4	KNOWLEDGE HIERARCHIES	53
4.1	Today Pittsburgh, Tomorrow The World	54
4.2	Knowledge Used by ARGOS	56
Chapter 5	RESULTS	58
5.1	Input	59
5.2	Experiments	60
5.2.1	Use of Location and Shape in Network Reduction	60
5.2.2	Determination of Network Reduction Amount	61
5.2.3	Rejection of Size Knowledge	62
5.3	Unsegmented System: Results	62
5.4	Pre-Segmented System: Results	64
Chapter 6	CONCLUSION	67
6.1	Error Analysis	67
6.1.1	Scale Errors	68
6.1.2	Position Errors	68
6.1.3	Shape Errors	69
6.1.4	Future Expansion	70
6.2	Summary	70
6.3	Discussion	72
6.4	Conclusion	73
Appendix I	Original Photographs	A1
Appendix II	Primitive Picture Elements	A5
Appendix III	Original City Map	A8
Appendix IV	Digitized City Map	A9
Appendix V	Generated City Views	A10
Appendix VI	Human and ARGOS Labeling	A15
Appendix VII	ARGOS Labeling of Hand Segmented Test Image 5	A30
Appendix VIII	ARGOS Labeling of Machine Segmented Test Image 5	A31
References		A32

ABSTRACT

ARGOS is an image understanding system. It builds a three-dimensional model of the task domain and uses hypothesized two-dimensional views of the model to label images. It currently achieves less than 20% error by area when labeling real-world (city of Pittsburgh) photographs with a knowledge base of over fifty objects. In addition, the system can determine the angle of view around the city with approximately 40 degrees of error.

The labeling technique used by ARGOS is called Locus search. Locus is a non-backtracking graph search technique in which a beam of near-miss alternatives around the best path are extended in parallel through the graph. After the graph has been searched in breadth-first order, the beam of possibilities is examined in reverse order to extract a near-optimal path. This path defines a labeling of the image and is only sub-optimal because of the pruning heuristics used in the beam creation.

Locus search has been used in the interpretation of speech (Lowerre, 1976). Its implementation in the image understanding task requires major modifications due to the non-linearity of the signal. Instead of implementing a form of a first-order Markov search which relies on only one previous node in the beam (as the speech system does), ARGOS implements an "adjacency first-order" Markov system that relies on *all* surrounding nodes in the physical image.

This thesis formulates image understanding as a problem of search; shows how Locus search can be used to label images; describes the many sources of knowledge used in the interpretation; shows how knowledge represented as a network can be used to constrain the search; explores extensions to the use of knowledge; and presents the experimental results of ARGOS. Its main contributions are the demonstration that Locus search can be used for image understanding and the exploration of issues involved in this use.

ACKNOWLEDGEMENT

Many researchers use this space to thank the countless and otherwise nameless people who have helped in the completion of their thesis. Parents, wives, and children are high on the list along with fellow researchers. But this very important space is often used to thank first-grade teachers, janitors, and summer camp counselors. Occasionally you get a guy who thanks his rabbit.

In my case, there is only one person to thank because that person's efforts were of major significance in producing this thesis. To thank anyone else would reduce the importance of this person's efforts. He was responsible for every stage of my graduate education and will command my respect for years to come. And so, before moving on to the thesis, I have only one thing to say. Thank you, Raj.

CHAPTER 1: INTRODUCTION

This thesis is about ARGOS, a computer program that can understand images. The term "image understanding" means that the program can identify the major components of a scene by using knowledge about the structure of the scene. For example, ARGOS is able to use a map of downtown Pittsburgh to aid in the identification of the major buildings, rivers, etc. in photographs of the city.

The technique that ARGOS uses to identify, or label, these images is called Locus search. Locus is a powerful search technique that uses a recursively defined evaluation function to scan an image while using networks of knowledge about the scene to constrain the search. After the image has been scanned once, Locus is able to extract picture labels from the results of the search.

One of the significant features of Locus search is its ability to use many diverse sources of scene knowledge when labeling images. Not only is it able to use information about the positions of the buildings, but it also knows their size, shape, and environment. In general, it is expected that Locus search can be a useful technique in applications where knowledge is used to attain goals.

This thesis, therefore, has two contributions. First, it presents a detailed discussion of Locus search as it applies to the image interpretation problem. Some of the new issues that arise from this implementation are the modification of the Markov assumption used by Locus and the hierarchical application of knowledge networks. These changes are necessary due to the topological complexity of the signal (i.e. images are two-dimensional) and the immense amount of knowledge needed by a complex vision system. The second and more significant contribution is ARGOS: a working image understanding system that can deal with real-world scenes. ARGOS is also important because it works from an internal three-dimensional model of the task environment.

1.1: IMAGE UNDERSTANDING

Image understanding is defined as the application of knowledge to the task of interpreting an image. This application consists of finding a match between the knowledge and the image. Once the match has been made, the image is both interpreted and understood. It is interpreted because it has been linked to the knowledge and therefore is labeled with knowledge-based descriptions. The image is also understood because it is possible to extrapolate the interpretation knowledge.

For example, ARGOS has knowledge about Pittsburgh. To understand a new image, it must find a match between that image and the Pittsburgh knowledge. This match will consist of pairings that link areas of the image with objects in the knowledge (i.e. "the area at the top is sky, the area at the bottom is Monongahela River, the area to the left of center is Hilton Hotel, etc."). With the results of this match, it is possible to delve farther into the knowledge and make statements that are not explicitly in the image ("this view is from Mount Washington"). It is this ability that makes ARGOS an image understanding system.

In general, the image interpretation process has three components: the image, the knowledge, and the match. The image is often refered to as the signal and the knowledge is the symbol. The match is therefore a signal-to-symbol match. The next section examines a number of image interpretation systems in these terms.

1.2: OTHER SYSTEMS

The table below lists a few image interpretation and image understanding systems in terms of their signal, symbol, and match. Each system is described more fully in the next sections.

Who	Signal	Symbol	Match
Barrow and Popplestone	Segment	Relational Network	Heuristic Search
Fischler and Elschlager	Segment	Relational Network	Linear Embedding (Dynamic Programming)
Feldman and Yakimovsky	Pixel	Relational Probabilities	Best-First
Tenenbaum and Barrow (IGS)	Pixel	Relations	Relaxation
Sakai et. al.	Segment	Semantic Network (Knowledge Blocks)	Data-driven Search

Keng and Fu	Bit Pattern	Grammar	Bottom-up Parse
Perkins	Lines and Curves	Lines and Curves	Brute Force
Waltz	Lines and Shadows	Junction Dictionary	Relaxation
Mackworth (MAPSEE)	Lines	Relational Network	Relaxation
Ballard et. al.	Lines, Segments, Pixels, etc.	Semantic Network	Arbitrary
Williams et. al. (VISIONS)	Segments	Semantic Network	Best-First
Rubin (ARGOS)	Pixel or Segment	Relational Network	Beam Search

1.2.1: Barrow and Popplestone

Barrow and Popplestone (1971) divide their images into segments. A coffee cup, for example, is broken down into three segments: the outside of the cup, the inside of the cup, and the hole in the handle. These three segments compose the signal description of the cup.

The symbolic knowledge consists of a relational network. Nodes in the network are region names and arcs that connect nodes are "facts" about the regions. For example, a knowledge network that describes the previously mentioned coffee cup will have three nodes for the inside, outside, and hole regions. The arc which connects the outside region and the hole region will contain the fact that the outside region completely surrounds the hole region. In addition, each node contains descriptive knowledge which enables the system to help identify the proper segment (i.e. the hole region is a small and highly compact segment).

The most obvious match process requires the evaluation of all of the segments against each node in the knowledge network. Even with only three segments and three nodes there is a possibility of $3^3=27$ different ways to interpret the image. Therefore the match technique is important not only for accuracy but also for space and time savings. Barrow and Popplestone use the heuristic search technique which views the space of segment-node alternatives as a search tree. It then searches this tree using details from the knowledge network to guide the search. Assume that the "trunk" of the search tree branches three ways to indicate the three pairings: (segment 1 is inside), (segment 1 is outside), and (segment 1 is hole). Each of these branches divides three ways to list the possibilities for segment 2 and each of the nine segment 2 branches splits three more ways for the segment 3 options. If, however, segment 2 is surrounded by segment 1 in the image, then the search process need not examine all

nine possibilities for segment 2, just those possibilities which satisfy the knowledge constraint of containment. Thus, heuristic search is able to prune large portions of the search tree by examining only the match possibilities suggested by the symbolic knowledge.

The disadvantage of heuristic search is that it still must examine all feasible possibilities. Locus search goes one step further in its pruning by evaluating each path and examining only the most likely of the feasible alternatives.

1.2.2: Fischler and Elschlager

Fischler and Elschlager (1973) also use image segments and relational knowledge networks. The networks consist of numerical "springs" which connect region nodes. These springs must be compressed and expanded by the match process to find the overall set of labels that has the least spring "tension". The match process uses a form of dynamic programming (Bellman and Dreyfus, 1962) which does not suffer from the combinatorial explosion of tree searching. Their match, called the Linear Embedding Algorithm, prunes all but a constant number of search tree nodes at each level. This match may fail to find the globally optimal interpretation, but it is efficient. In addition, it allows for noisy data by accepting highly "stretched" springs in the final labeling.

The reduction of the search space to a linear growth is the same philosophy used by Locus. The difference is that Locus uses a dynamic threshold to admit the most likely nodes within the fixed growth limit. Therefore, growth of the search space is variable but linearly bounded.

1.2.3: Feldman and Yakimovsky

Feldman and Yakimovsky (1974) use relational probability knowledge to guide the isolation and identification of segments. The image starts as a collection of picture elements (pixels) which are organized as a rectangular grid of points. The system combines pixels into segments by breaking down the boundaries between areas of the image that are similar. It uses a probabilistic utility function of the signal and the symbol to compute the most probable boundary to break. This best-first control structure proceeds until a stopping criteria is reached at which time the image is segmented and labeled. The most interesting aspect of this system is its use of Bayesian decision theory in computing the utility function.

Locus does not explicitly segment images. It labels each point purely on the basis of context knowledge. The final labeling may be used to define a segmentation by observing groups of similar labels, but that is only a by-product.

Another difference between this work and Locus is formality. Locus does not follow the rules of decision theory very closely: it prunes and approximates frequently. The result is a pseudo-statistical system that is only formal within heuristic bounds.

1.2.4: Tenenbaum and Barrow

Tenenbaum and Barrow (1976) apply relational knowledge to the segmentation and interpretation task. Their system starts by crudely segmenting on the basis of pixel similarity and then making a set of labels for each segment. It then applies iterative relaxation to get a consistent label a refined segmentation of the image. Relaxation works by repeatedly evaluating the junctions of segments and revising segment borders and labels on the basis of relational constraints. Each pass of the relaxation operation revises the list of possible segment labels on the basis of the surrounding segment label lists. For example, if a large segment with the label options wall and door surrounds a small segment with the label options window and doorknob, and one iteration of relaxation eliminates the door option, then the next iteration will eliminate the doorknob option. Thus, global knowledge can propagate through the image as the relaxation iterates. The final labeling is obtained when the relaxation converges and yields no further change to the interpretation.

The problem with relaxation techniques is they do not guarantee convergence and so do not have well defined termination criteria.

1.2.5: Sakai, Kanade, and Ohta

Sakai et. al. (1976) label image segments using a sophisticated knowledge network that can contain procedural code and arbitrary knowledge constraints. These "knowledge blocks" are applied first to key segments in the image. Since the knowledge blocks can have control sections, they do the bulk of the search. Thus, instantiation of a knowledge block can lead to the hypothesis, verification, and rejection of other knowledge blocks. The control procedure simply runs through the list of untested blocks until there are no more, at which time the interpretation is complete. Although this system is fairly sophisticated, it suffers from the complexity of its data which must be carefully constructed.

1.2.6: Keng and Fu

Keng and Fu (1976) have a system which interprets satellite images by matching the binary patterns in windows of the image to templates in a grammar-style knowledge

base. Each window of 8x8 points in the image is compared with every known template in the grammar. Windows which evoke multiple templates are reduced with the grammar to a unique label. If, for example, a window contains two intersecting lines, each of which matches a highway template, then the grammar will generate a highway intersection interpretation for the window. The system uses no global constraining and relies heavily on its knowledge of the sensor characteristics of Landsat satellites.

1.2.7: Perkins

At General Motors, Perkins (1977) identifies parts on a conveyor belt by comparing their outlines to those in the knowledge base. Outlines consist of curves and straight lines which he calls concurves. A scoring algorithm counts the number of concurves in the image which match those in known parts. Since the system looks for relative scores, it can accept noisy data and occluded objects. The system is even able to extrapolate on partial matches and predict the complete placement of occluded parts and parts that are only partially in the field of view. However, the task domain is limited and the system would suffer from combinatorial explosion if it handled complex scenes. This is because it uses no heuristics to limit the search space: identification requires the comparison of all concurves in the image with all concurves in the knowledge base.

1.2.8: Waltz

Waltz (1975) matches knowledge in a list of rules to an image that consists of lines. In addition, each segment enclosed by lines can be shaded (a shadow) or unshaded. The knowledge base contains alternative interpretations for the various line junction types. Match is done with a form of relaxation that iteratively compares junction interpretations at the ends of each line segment. After the system has converged, the junction interpretations are used to label the surfaces and explain the shadows.

The system works in the blocks-world domain where perfect lighting makes shadows trivial to distinguish. ARGOS uses real-world scenes which are full of shadows that are hard to detect. However, the goal of Waltz's work is to explain shadows whereas the goal of ARGOS is to explore a search technique, so there is no fair basis for comparison.

1.2.9: Mackworth

Mackworth (1977) uses relaxation on a relational knowledge network. His input signal

is a "sketchmap" which consists of chains of lines. Some chains are closed and form lakes and islands. Others are open and form rivers and roads. The system initially makes many weak pairings of labels to chains and then refines the pairings using a network consistency algorithm. This algorithm simply re-evaluates the pairing options based on each chain's environment. The system eventually converges on an interpretation of the image. It is interesting to note that an initially noisy image can be interpreted many times by using the final results of the first interpretation to refine the sketchmap. The entire process is then repeated for a better overall interpretation.

1.2.10: Ballard, Brown, and Feldman

Ballard et. al. (1977) use a semantic network that describes the scene at several levels (region relations, pixel adjacencies, line lengths, etc.). Each image is also described at many levels as a collection of lines, segments, volumes, etc. The user of the system codes a query procedure using a control structure of his own choosing which explores the semantic network and builds an instantiation of the network that corresponds to the image. This instantiation, called a "sketchmap" (not to be confused with Mackworth's sketchmaps) is an interpretation that links parts of the image to parts of the semantic network. An example of a query procedure that has been built is a rib matcher for chest X-rays. This procedure uses cost functions to numerically determine the best match of signal-to-symbol. Although it would be possible to code Locus in this framework, very few alternative search strategies have actually been explored.

1.2.11: Williams, Lowrance, Hanson, and Riseman

Williams et. al. (1977) describe an ambitious effort to apply multiple sources of knowledge to a segmented image. Their knowledge base is a semantic network of knowledge sources that are linked hierarchically in the same manner as the Hearsay II speech system (Erman and Lesser, 1975). It deals with the image on many levels of representation including lines, segments, and frames (Minsky, 1975). Control is divided among the many knowledge sources which repeatedly hypothesize and verify signal-to-symbol matches in a best-first search order. There are even knowledge sources that guide other knowledge sources by focusing the system's attention.

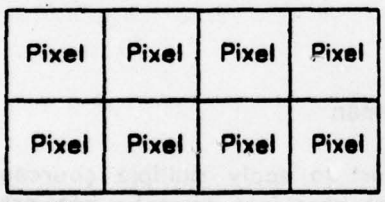
Locus attempts to unify many knowledge sources into one so that a single control structure can be applied. It rejects the notion that multiple sources of knowledge must be used independently.

1.3: LOCUS SEARCH

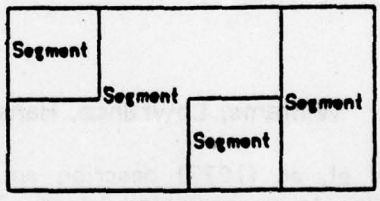
ARGOS uses Locus search to interpret images. Locus was first used in the Harpy speech understanding system (Lowerre, 1976). Harpy uses this technique on the recognition of spoken utterances and is currently the best speech recognition system in existence. Prior to Harpy, Dragon (Baker, 1975) used a breadth first graph searching technique to attain the same goal.

Dragon is the theoretical grandfather of ARGOS. It uses a probabilistic function of a Markov model to find an optimal path through a knowledge network. The technique is very similar to the Viterbi Algorithm (Forney, 1973). Harpy relaxes the formality of the probabilistic function and demonstrates that heuristics can be used to improve the search. ARGOS goes one step further by modifying the Markov assumption so that the multi-dimensionality of images can be handled. Instead of requiring that a first-order Markov system rely on only one previous node in the search tree, ARGOS implements an "adjacency first-order" Markov system that relies on *all* surrounding nodes in the image.

Before proceeding, it is useful to stop for a moment and examine ARGOS in the light of the signal-to-symbol match paradigm of the previous section. In Locus, the symbolic knowledge is composed of units called Primitive Picture Elements (PPEs). These PPEs are organized into relational networks which specify knowledge about a scene. In addition, PPEs have signal characteristics associated with them so that they may be directly compared with parts of the image. ARGOS does not place any constraint on the nature of the signal, which is why it is able to interpret both pre-segmented images and unsegmented images.



Unsegmented Image



Pre-segmented Image

Although the signal is frequently refered to as a pixel, it should be understood that ARGOS can deal with either pixels or arbitrarily shaped segments.

A simple example of PPE selection is in order. Assume that Locus must match a satellite image of a field to a knowledge base which contains information about crop locations. Locus would define each crop to be a PPE so that a relational network could be built describing the crop locations. Similarly, each of the crop PPEs would have signal characteristics that allow it to be matched with the image (i.e. "the alfalfa PPE registers 4 on a brightness scale of 1 to 10"). Thus, PPEs are the common ground between signal and symbol.

The match aspect of Locus is a two-pass search process that explores the space of signal-to-symbol pairings. The first pass of the search, called the forward pass,

constructs a highly pruned tree of alternative pairings. Each level of this search tree corresponds to a different pixel in the image, and the various nodes at each level are the alternative PPE labels for that pixel. The selection of tree entries (which are likely PPEs for a given pixel) is based on a recursively defined evaluation function. This function uses knowledge from the network, the image, and the immediate environment of the search tree to determine a uniform evaluation of a signal-to-symbol match. The completed search tree contains a multitude of pixel-to-PPE matches, *each of which identifies a path of "optimal" pixel-to-PPE matches above it in the tree.* The second pass of Locus search, called the backtrace, simply re-examines the search tree from the bottom up, gathering the optimal pixel-to-PPE matches into a unique labeling of the image. Because of the Markov nature of the forward pass, the backtrace is able to bring global constraint to bear in the final selection of image labels.

It is interesting to note that the search tree built by the forward pass is so highly pruned that it resembles a varying sized beam of alternatives and is sometimes called the "beam". The beam is functionally equivalent to the stack of alternatives generated by standard backtracking search algorithms: they both list the options that are under consideration. However, beam search is superior to backtracking methods because it avoids thrashing behavior.

1.4: ARGOS

ARGOS interprets pictures of downtown Pittsburgh. Fifteen of these pictures, reproduced in Appendix I, were taken from five vantage points around the city. To enhance variability, some were shot with a standard 50mm. lens and others were taken with a 28mm. wide-angle lens. Seven were used for training and the other eight were saved for test purposes. In the following table of images, the column labeled "Number of Objects" is the number of different regions that were identified during human labeling.

Image	Vantage Point	Lens	Number of Objects
Training 1	Northwest	28mm.	17
Training 2	Northwest	28mm.	19
Training 3	Northeast	28mm.	17
Training 4	West	50mm.	26
Training 5	West	28mm.	27
Training 6	Downtown	28mm.	9
Training 7	Southwest	28mm.	27
Test 1	Northwest	50mm.	16
Test 2	Northwest	28mm.	17
Test 3	Northwest	50mm.	15
Test 4	Northeast	50mm.	15
Test 5	Southwest	50mm.	13
Test 6	Southwest	50mm.	19

Test 7	Southwest	50mm.	21
Test 8	Downtown	28mm.	7

Each image was originally digitized into a rectangular grid of 700 pixels across and 525 pixels down. However, ARGOS interprets reduced versions of those images that are 75 by 100 pixels in size. It can also interpret images that have been divided into arbitrarily shaped segments.

The system runs on a PDP-KL10 computer and requires approximately 100,000 words of 36-bit memory. The largest part of this space is used to store the parallel search paths (typically 25 paths). It takes about five minutes of processor time to label one 75 by 100 image, but less than a minute to label a pre-segmented image.

1.5: READING THIS THESIS

The next chapter discusses Locus search in great detail. It starts by explaining how image understanding can be formulated as a problem of search. From there, it describes the organization of the knowledge networks that are used in ARGOS. Following that is a discussion of the low-level system in ARGOS: the techniques used to give signal characteristics to PPEs. Finally comes a detailed explanation of the search and an example of the entire process.

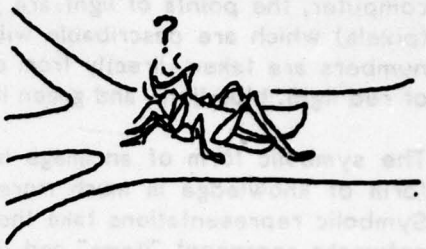
Chapter 3 discusses how many different knowledge sources can be used with Locus search. In addition to expounding upon object adjacency and image pre-segmentation, the chapter discusses the use of object size, shape, and location.

Chapter 4 discusses how knowledge can be organized hierarchically to infinitely expand the power of Locus. Most of this chapter is speculation, but it does conclude with a review of where ARGOS stands in its hierarchical use of knowledge.

Chapter 5 describes the initial experimental results that were obtained with ARGOS. The system is currently able to label the fifteen city scenes with 20% error at the pixel level. It can also determine the angle of view around the city with an average error of 40^{o}. Considering the complexity of the knowledge and the images, this is quite good.

Chapter 6 concludes the thesis by evaluating the results of ARGOS.

CHAPTER 2: LOCUS SEARCH



This chapter describes the Locus model of search as it is used in the Image Understanding task. Locus is an efficient, non-backtracking search technique that has proved itself successful in the Harpy speech understanding system (Lowerre & Reddy, 1977). It is able to provide near-optimal solutions to the network search problem in time that is linearly bounded by the complexity of the input signal.

This chapter discusses the search technique in detail. The first section describes the representation used by Locus to describe images and knowledge networks.

The next section is a discussion of the knowledge networks that are employed. Coupled with this discussion is an explanation of why image interpretation is treated as a problem of search.

Following that is a discussion of the low-level processing that is done in ARGOS. This thesis is not specifically concerned with the low-level aspect of image interpretation, so the discussion is somewhat cursory.

An explanation of the search process follows next. Since application of the search technique to image interpretation is one of the main contributions of this thesis, the discussion is quite long and contains many examples.

The chapter closes with an example of Locus search.

2.1: STRUCTURE OF LOCUS KNOWLEDGE

Image understanding is the application of knowledge to the task of interpreting images. As Chapter 1 pointed out, this process requires the matching of two forms of data: the sensed data which is the signal and the knowledge structure which is the symbol. The

signal is the raw form: points of light pieced together to comprise the image. For a computer, the points of light are grouped together into segments or picture elements (pixels) which are describable with a set of numbers called a feature vector. These numbers are taken directly from camera sensors and tell such values as the intensity of red light, blue light, and green light.

The symbolic form of an image is the knowledge representation of the scene. This form of knowledge is much more compact than the signal, so it is easier to store. Symbolic representations take the form of networks of information: the nodes of the networks represent "items" and the inter-connections of the nodes represent "facts" or qualifications about the items.

Before a machine can match a signal to a symbol, it must be able to compare the two forms. Some systems use a third representation to do this (Ballard et. al., 1977). ARGOS does just the opposite: instead of creating a new representation, it combines the existing ones so that the signal and the symbol are all describable with the same units. These units are called Primitive Picture Elements.

The Primitive Picture Element, or PPE, is the basic building block of both the signal and the symbol. Every sensed image can be described with PPEs since each pixel can be given a unique PPE label. Similarly, the general knowledge of a scene (sky above river under bridge, etc.) can be described with a network, all of whose nodes are PPEs. PPEs can be thought of as the smallest units of representation that exist for the micro-world of the image task. Alternately, they can be thought of as the *largest* object that is *both* homogeneous to the signal *and* homogeneous to the symbol. For example, look at training scene 6 in Appendix I (page A2). The building in the center is the Pennsylvania State Office Building and it has a lobby that looks much different than the rest of the structure. Proper identification of this building would therefore require two PPEs called State-office and State-office-lobby because, although they are symbolically homogeneous (both State Office), they are not homogeneous to the signal (they look different). Similarly, observe the three cross-shaped buildings, one of which is obscuring the other two, on the right side of in test scene 2 (page A3). Although all three look the same, they are assigned different PPEs because they are symbolically different (from left to right, they are Gateway Two, Gateway One, and Gateway Three). To sum it up, the PPE is the label that Locus uses when interpreting images.

The choice of PPEs varies with the micro-world being explored. It is dependent on the level of detail in the knowledge base and on the ability of the system to optically distinguish different parts of an object. If, for example, one wishes to determine whether an image is a city scene or an office scene, then neither the sensed data nor the symbolic representation need be very complex. The PPEs sky, building, road, wall, desk, and floor would suffice. Note that each of these PPEs is comprised of optically similar pixels in the sensed image and represents adequate symbolic information to determine the scene type.

If, however, a finer level of scene detail were to be interpreted, then the selection of PPEs would have to be more complex. The following PPEs might be selected for the detailed analysis of a Pittsburgh city scene:

Hilton awning	Commonwealth Place
Hilton conference area	Liberty Avenue East
Hilton elevator shaft	Liberty Avenue West
Hilton main structure	Liberty Avenue Island
Hilton windows	Blvd. of the Allies
Hilton doors	Mountains to the North
Allegheny River	Mountains to the south
Sky	Park

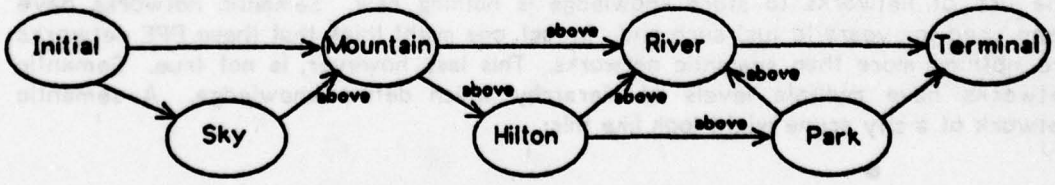
etc.

For a more complete discussion of PPE selection as it relates to knowledge hierarchies, see Chapter 4.

Given that a set of PPEs is to be used to interpret an image, a number of questions immediately come to mind. Two obvious ones are how the sensed data is to be compared with PPEs and how the PPEs are to be organized as a symbolic knowledge base. More important is the issue of how these two forms are matched. Sections 2.2 and 2.3 discuss how PPEs are organized symbolically and optically. Section 2.4 describes the match process, which of course is Locus search.

2.2: SYMBOLIC USE OF PPEs

Imagine that a network is constructed whose nodes are all PPEs. The network contains an initial node and a terminal node between which all other nodes are somehow connected. Each node is an item in the network and each node connection is a fact about the two items it connects. Therefore, a complete network is a set of facts about PPEs and is called a knowledge network. In addition, a complete path from the initial node to the terminal node is a collection of knowledge that is consistent (i.e. all facts agree with each other). The network might look like this:



Imagine further that for every image which is M by N pixels in size, there exists a path of MxN nodes stretching from the initial node to the terminal node which forms a

consistent "statement" about that image[1]. The existence of a PPE node in a given position along the path is equivalent to a label assignment of that PPE's symbolic name to the pixel in that path position. The path, therefore, forms a symbolic description of the image, taken from the knowledge network.

Since a PPE path corresponds to a set of labels for an image, the image interpretation problem reduces to a search task. Knowledge for this search task comes from the network connections which guide the path selection. For example, assume that there are 50 PPEs in a network and that each one is connected to all of the others, yielding 2500 node connections. This represents no constraint on the network paths and therefore no knowledge. The network would then consist of an initial node, a terminal node, and one node for each of the 50 PPEs. If a sensed image which is 75 by 100 pixels in size is matched to this network, then the path through the 50 PPEs can take any of 50 choices at each step through the image. The number of different paths through the network would be 50^{7500}. If one of the 2500 node connections is removed, then the number of possible network paths is reduced. Knowledge, therefore, appears in the form of constraints on the interrelations of PPEs.

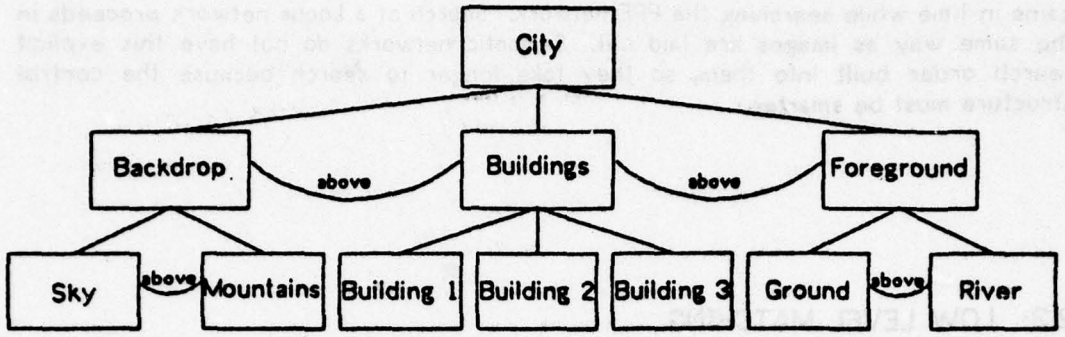
How are these constraints selected? One way is to build up the set of interrelations from a set of "expected" images. Since each network path corresponds to a legal image, the converse must also hold: each legal image has a unique network path. It is this converse form which defines how knowledge is encoded as a set of network constraints. Each image in the expected micro-world is combined into an overall knowledge network. If the expected images are all of downtown Pittsburgh, a city which is surrounded by mountains, then all images of the city are going to have a backdrop of mountains below the backdrop of sky (except for pathological viewpoints which can be ignored in this example). This simple piece of knowledge manifests itself directly as a network constraint: the sky PPE may not adjoin any of the building PPEs because the mountain PPE intervenes.

In addition to specifying legal adjacencies, the network arcs can contain information about the kind of adjacencies. An arc can specify that the PPEs which it connects are vertically or horizontally adjacent. It can also specify more complicated relationships such as "containership" and some other knowledge such as object size. However, as the discussion of shape in Chapter 3 will show more fully, most complex relationships can be built from a series of simpler constraints.

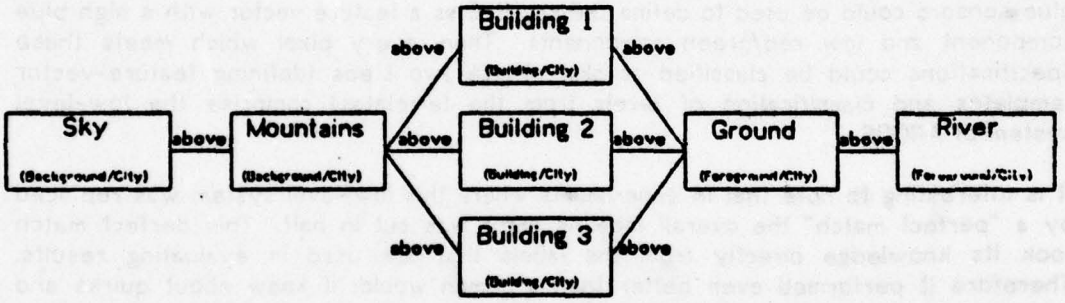
The use of networks to store knowledge is nothing new. Semantic networks have been used for years to just such end. In fact, one might think that these PPE networks are nothing more than semantic networks. This last, however, is not true. Semantic networks have multiple levels of hierarchy which define knowledge. A semantic network of a city scene might look like this:

---

1 These MxN nodes can be thought of as being linearly connected into a string of nodes, but in the 2-dimensional world of images, the topology is not that simple. This is because the points in an image are not attached to each other in a single line.



In the above semantic network, all depth relationships (such as the one between backdrop and sky) are "within" by default and all nodes at a given breadth level (such as the one between backdrop and buildings) are unrelated unless specified. PPE networks however, have no hierarchy of "within". All relationships must be specified explicitly. A Locus version of the above network would be represented with the seven PPEs taken from the lowest level of the semantic network. Notice that the layout of the PPE constraints implies that the top of the image is on the left "side" of the network:



The PPE network differs in two ways from the semantic network. First, all information about a node is encoded at the node site, and not at some "higher level" of the network. This is because there is only one level of the network. It might seem that this causes each node to use excessive amounts of storage, but in practice, it only requires a few extra pointers at each node. The second difference between PPE networks and semantic networks is size. Since semantic networks can share low-level descriptions, they can often save space with common sub-networks. PPE networks cannot do this because of their uni-level structure. As a result, PPE networks tend to be quite large. This is one of the reasons that world knowledge is broken down into multiple networks instead of being placed into one knowledge structure (see Chapter 4).

Why are PPE networks structured so as to make them excessively large? The answer is found in the classical tradeoff of space for time. Single-level networks are much easier to search and PPE networks are no exception. What Locus loses in space, it

gains in time while searching the PPE network. Search of a Locus network proceeds in the same way as images are laid out. Semantic networks do not have this explicit search order built into them, so they take longer to search because the control structure must be smarter.

2.3: LOW LEVEL MATCHING

Before the search process can be discussed, there is one more background issue that must be covered: the direct comparison of the sensed data to PPEs. This comparison is at the low-level end of vision research and is critical to any high-level system such as ARGOS. However, since this thesis is concerned only with the high-level end, the discussion presented here contains no new research. In fact, it does not even pretend to represent the optimal techniques.

Implicit in the definition of a PPE is the iconic homogeneity factor: every pixel[2] that belongs to a PPE class has similar sensor parameters. For example, the red, green, and blue sensors could be used to define the sky PPE as a feature vector with a high blue component and low red/green components. Then, every pixel which meets these specifications could be classified as sky. These two steps (defining feature-vector templates and classification of pixels from the templates) comprise the low-level system of ARGOS.

It is interesting to note that in experiments where this low-level system was replaced by a "perfect match" the overall labeling error was cut in half. This perfect match took its knowledge directly from the labels that are used in evaluating results. Therefore it performed even better than a human would: it knew about quirks and errors in the evaluation process. However, this experiment still served to show that a better low-level system would improve ARGOS results.

2.3.1: Feature Vector Templates

In vision systems today, there is a glut of information available to describe an image. The images that ARGOS works with come digitized as 525x700 points of light. Each point is described by three 8-bit values from red, green, and blue sensors. Using red, green, and blue, it is possible to obtain hue, intensity, and saturation, or, as the color television industry in America does, obtain Y, I, and Q (Price, 1976). Both of these derived forms cover the color spectrum. Every point of light, therefore, is available as 9 different sensor values.

---

2 Recall that the term "pixel" refers to any area of the image on which ARGOS operates, including arbitrarily shaped segments.

To simplify labeling, ARGOS does not work with 525x700 images. Not only is it prohibitively expensive to search trees with over 300,000 depth levels, but the interpretation being sought does not require such detail (see the list of PPEs in Appendix II for an understanding of the desired level of detail). ARGOS reduces each image before labeling. When the system is working with segmented images, it typically divides the image into as many as 100 arbitrarily shaped segments (see Appendices VII and VIII). When the system is labeling a regular grid, each image is reduced by a factor of 7 in each dimension before being processed. This reduction to 75x100 images makes the search more manageable. It is the largest reduction that can be performed without damaging a human's ability to recognize the images at the desired level of detail. In addition, the 7x7 window allows the texture operators (described below) to distinguish many of the buildings from each other.

The selection of feature vector templates now has two aspects: the choice of sensors and the choice of reduction operations to perform on the points of light in each window. The reduction operators that were explored were: mean, mode, median, standard deviation, Z.C.C. (zero crossing count: a micro-edge detector from Price, 1976), and contrast (a function of the 4th moment of the distribution curve from Tamura et. al., 1977). The list could go on forever but this thesis cannot. With 9 sensors and 6 possible operators on each sensor, the feature vector has a potential size of 54. Space limitations in ARGOS reduced that number to 6.

There is no adequate justification for the limit of 6 elements in the feature vector. For that matter, there is no proof that 6 are needed. The selection of the feature vector components was done experimentally by evaluating the labeling quality of each component on a training image. Although more formal techniques exist (Tou and Gonzalez, 1974) this method was chosen for simplicity. It was also felt that more robust selection criteria were not needed in the non-production environment of this thesis.

Feature vector selection proceeded as follows: a training image was labeled by an unbiased person, feature-templates were generated for each sensor/reduction pair, then each feature-template was tested on its ability to reproduce the initial labeling. The table below shows the percentage of the image that was labeled correctly for each reduction operator and each sensor. Note, for example, that the mean value from the blue sensor was the most accurate because when each PPE was classified solely in terms of that template, 44% of the label assignments made were correct.

	Z.C.C.	Mean	S.D.	Mode	Median	Contrast
Red	29%	22%	30%	22%	29%	9%
Green	27%	22%	31%	24%	36%	8%
Blue	22%	44%	40%	37%	42%	8%
Hue	23%	7%	18%	27%	42%	11%
Intensity	26%	23%	31%	26%	36%	8%
Saturation	24%	23%	24%	21%	19%	7%
Y	31%	21%	30%	22%	35%	8%
I	30%	20%	26%	17%	28%	11%
Q	22%	30%	37%	31%	29%	7%

From the above table, plus additional information about how well each sensor did for various parts of the scene, six feature vector elements were chosen. It was decided first that since Hue/Intensity/Saturation and Y/I/Q are simple re-combinations of Red/Green/Blue, the former could safely be rejected in favor of the latter. This is also supported by the fact that humans perceive color as Red/Green/Blue so ARGOS should do the same if it wants to best mimic human perception. The next choice is among the operators. Median, a good low-texture operator, performs best according to the above table. Since city scenes have many high-texture objects, some other operator is also needed.

It is interesting to note that the statistics in the above table are of no help in selecting a high-texture operator since the test scene had much low-texture area, thus causing high-texture operators to score badly. In particular, it was noticed that contrast was best at labeling the high-texture areas, even though the above table indicates that it is the worst operator. Therefore, the final feature vector selection contains these sensors and operators:

Median of Red	Contrast of Red
Median of Green	Contrast of Green
Median of Blue	Contrast of Blue

One additional consideration must be given to the feature vector templates: they should be able to accurately describe objects that have multiple appearances in the training images. For example, assume that two images show a building alternately in the shade and in direct sunlight. A single feature vector for these two views would attempt to combine both appearances and, in so doing, destroy the ability of the feature vector to label either view. Although more sophisticated image operators could be used to distinguish shadows, the point here is that objects may vary in appearance and this variance must be handled properly.

ARGOS solves this problem by creating multiple feature vector templates for objects with multiple appearances. The decision to create an additional template is made automatically when the standard deviation of more than two feature vector elements exceeds 20% of their possible range. If, for example, the Median of Blue range is 100, and the standard deviation of five instances of a building is over 20 in the Median of Blue position, then that element of the feature vector is too diverse. If more than two elements are too diverse, the feature vector is split in the middle of the dynamic range of the most diverse element. When training on seven images, ARGOS generates as many as three feature vector templates for each object, although one template typically suffices.

2.3.2: Distance Metrics

Now that six elements have been selected for the feature vector, a distance metric must be found for the pixel-to-PPE likelihood calculation. A distance metric is simply a

way of determining the likelihood that a given pixel in the sensed image belongs to a PPE class. The metric provides what can be called an "optical match" between signal and symbol: it tells how close they are to looking the same. The simplest distance metric is subtraction where the optical match is based on the difference between the PPE's feature vector template and the pixel's feature vector. For example, if the sky PPE has the template values: (red median=0, green median=0, blue median=10, red contrast=0, green contrast=0, blue contrast=0) for the feature vector selected in the previous section (which indicates strong, solid blue) and a sensed pixel has the feature vector (1, 2, 7, 4, 2, 0) then the subtraction distance metric yields the vector (1, 2, 3, 4, 2, 0). An optical match can be computed using a percentage of the worst-case distance vector (which would be (10, 10, 10, 10, 10, 10) in this case).

ARGOS uses a slightly more complex metric that is commonly found in vision systems: weighted-Euclidean distance. Euclidean distance is, of course, the square root of the sum of the squares of the distances. In the above example, the Euclidean distance would be $[1^2 + 2^2 + 3^2 + 4^2 + 2^2 + 0^2]^{.5} = 5.8$. ARGOS goes one step further: each component of the feature vector template is weighted so that unimportant components will not drag the optical match value down. The weights are a six-tuple which might, for the sky example, be (.18, .18, .1, .18, .18, .18), indicating that the low-contrast (median) blue sensor is less constrained, but all others contribute equally to the likelihood calculations. In this case, the weighting allows the sky PPE to be deep blue or light blue, but no red, green, or high-texture. The weighted-Euclidean distance metric then becomes $[(1^2)(.18) + (2^2)(.18) + (3^2)(.1) + (4^2)(.18) + (2^2)(.18) + (0^2)(.18)]^{.5} = 2.2$.

The selection of weighting factors is another badly justified algorithm in ARGOS. The problem is that at the root of all selection is a human who must train the machine on each PPE. The training is done by interactively outlining examples of each PPE region. From these regions, the machine is told to select feature vectors for the PPEs. It begins its selection by computing the mean and standard deviation for all pixels in the region. Then, those pixels that are more than 1.5 standard deviations from the mean are rejected. This rejection of atypical pixels allows for human error in the labeling and prunes data points which may be unfairly contributing to the metric. The mean and standard deviation are then re-calculated and the adjusted standard deviation is used as a weighting factor for the adjusted mean (which becomes the template value). The use of standard deviation as a weighting factor on the mean is common practice because it is an indicator of the consistency of the data and therefore the usefulness of the mean value in classifying that data. The conversion of standard deviation to weighting factors is done by inverting the six standard deviation values for a given region and then linearly scaling them so that they sum to 1. Since feature vector components with high standard deviations indicate erratic quality on the part of that component, a high standard deviation will yield a low weight which will reduce the system's dependence on that component.

2.4: LOCUS SEARCH

There is only one step remaining in this labeling process: search. The knowledge environment has been formulated as a network and techniques have been devised for matching the sensed image to that network. The search process must now find the "correct" set of labels for the sensed image.

As the discussion of PPEs mentioned, the combinatorics of labeling an image are tremendous. ARGOS deals with images that are 75 by 100 pixels and contain a minimum of 50 PPEs. If no knowledge about region relationships is available, then the branching factor for a 50 PPE network is 50 (i.e. PPEs can branch 50 ways to all other PPEs) and the number of possible labelings is 50^{7500}, or about 10^{12750}. When knowledge is applied, the branching factor reduces significantly. For example, a typical 50 PPE network with knowledge has a branching factor of about four which reduces the number of possible labelings to 4^{7500}, or about 10^{4500}. Thus, the application of knowledge is the most powerful tool available in reducing the search space. No other technique can reduce the number of labelings by over 8000 orders of magnitude! However, there is still much reduction to be done before a single, optimal path is selected, because even 10^{4500} paths are too many for a computer to examine. Locus search is able to perform that reduction and extract a single labeling from the knowledge-reduced search space.

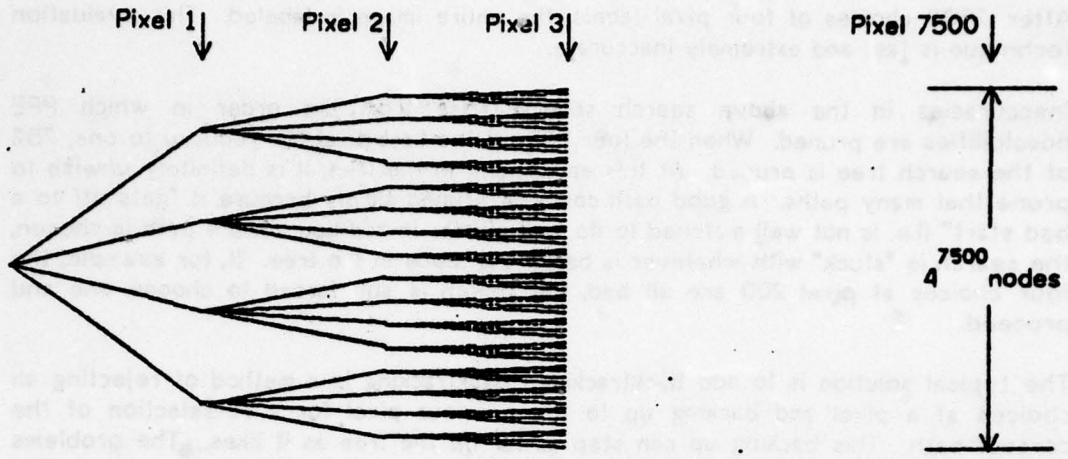
2.4.1: Introduction to Locus search

Finding the optimal path through a graph is a classical search problem with many alternative search strategies (Nilsson, 1971). What distinguishes ARGOS from other systems is its use of Locus search. Locus is a beam search heuristic in which all except a beam of near-miss alternatives around the best path are pruned from the search tree at each decision point. This reduces the exponential number of paths to explore without requiring backtracking or any iterative search.

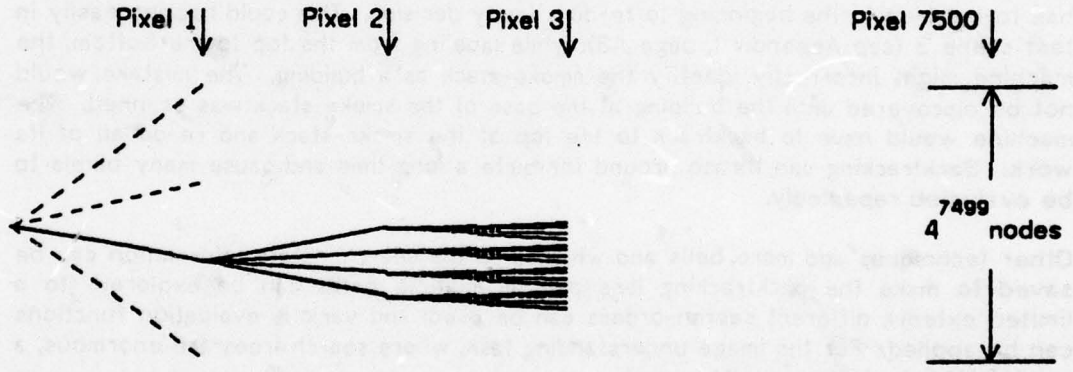
The remainder of this section describes the basic issues of search which are necessary to the understanding of Locus. More advanced readers will want to skip ahead in order to avoid sleep.

Search involves the creation of a search tree from the signal and the symbol. This creation is a simple expansion of the possible paths through a knowledge network. Just as each knowledge network path has a node for every image pixel, so every level of depth through a search tree corresponds to a different image pixel. Thus, a knowledge network node which is visited ten times in the labeling of an image will appear on ten levels of the search tree.

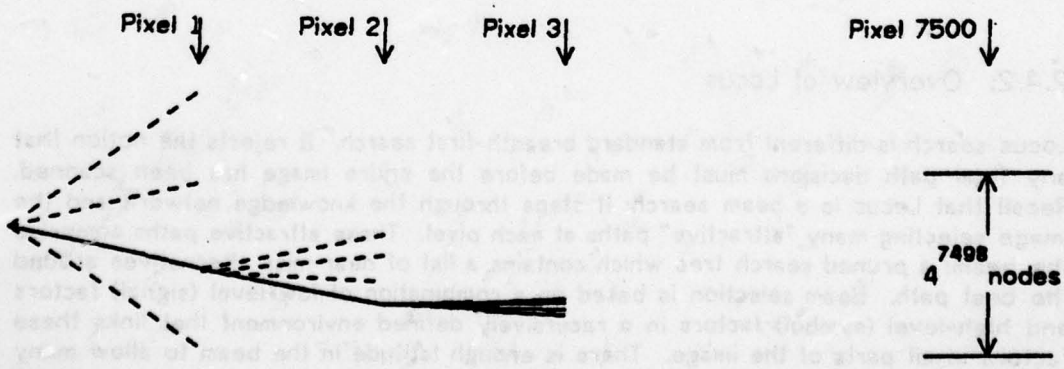
To fully understand Locus search, a description of some simpler search strategies is needed. Recall the 75x100 pixel image that is matched to a knowledge network with a branching factor of four. The search tree contains 4^{7500} paths which are laid out like this:



A single path must be selected which labels the image. A simple breadth ordered evaluation starts by examining the four possible PPEs at pixel 1 and selecting the best. The search tree then looks like this:



Next, the four choices at pixel 2 are evaluated and the search is reduced further:



After 7500 choices of four pixel labels, the entire image is labeled. This evaluation technique is fast and extremely inaccurate.

Inaccuracies in the above search scheme arise from the order in which PPE possibilities are pruned. When the four PPEs at the first pixel are reduced to one, 75% of the search tree is pruned. At this early point in the tree, it is definitely unwise to prune that many paths. A good path could be pruned simply because it "gets off to a bad start" (i.e. is not well matched to its first pixel). In addition, once a path is chosen, the search is "stuck" with whatever is below that node in the tree. If, for example, the four choices at pixel 200 are all bad, the search is still forced to choose one and proceed.

The typical solution is to add backtracking. Backtracking is a method of rejecting all choices at a pixel and backing up to the previous pixel for a re-selection of the correct path. This backing up can step as far up the tree as it likes. The problems with backtracking are numerous. First, how is it decided that all choices are bad? The machine doesn't know what's on the other paths so it can't be sure. It has already been established that the machine can't look at every point in the search tree, so some absolute measure must be available of the goodness of a path. Another problem with backtracking is time: the search might be near the end of the tree and find out that it has to back-up to the beginning to re-do a faulty decision. This could happen easily in test scene 3 (see Appendix I, page A3): while labeling from the top to the bottom, the machine might incorrectly identify the smoke-stack as a building. The mistake would not be discovered until the building at the base of the smoke-stack was scanned. The machine would have to backtrack to the top of the smoke-stack and re-do all of its work. Backtracking can thrash around for quite a long time and cause many pixels to be evaluated repeatedly.

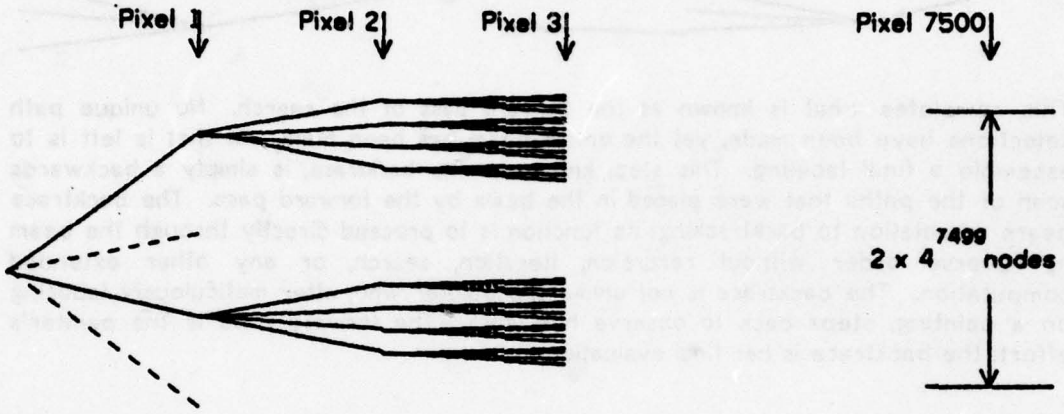
Other techniques add more bells and whistles to the search. State information can be saved to make the backtracking less painful; multiple paths can be explored (to a limited extent); different search-orders can be used; and various evaluation functions can be applied. For the image understanding task, where search trees are enormous, a much better technique must be used.

2.4.2: Overview of Locus

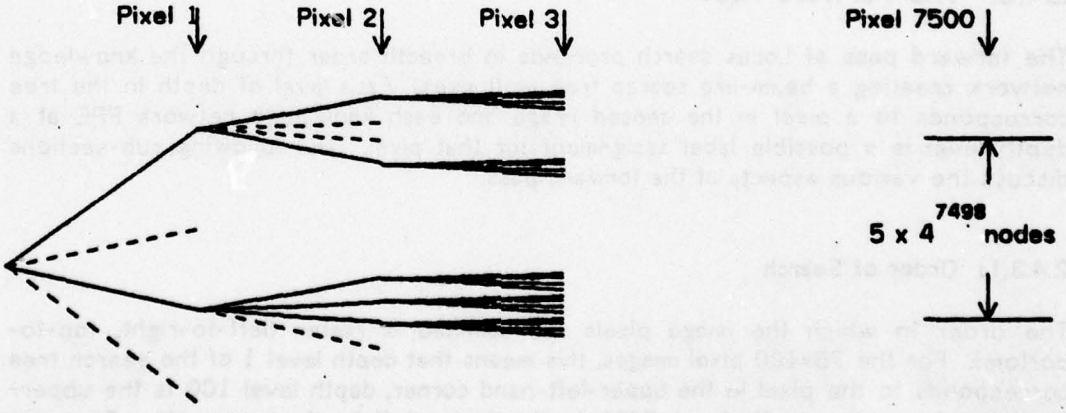
Locus search is different from standard breadth-first search. It rejects the notion that any final path decisions must be made before the entire image has been scanned. Recall that Locus is a beam search: it steps through the knowledge network and the image selecting many "attractive" paths at each pixel. These attractive paths comprise the beam: a pruned search tree which contains a list of near-miss alternatives around the best path. Beam selection is based on a combination of low-level (signal) factors and high-level (symbol) factors in a recursively defined environment that links these factors in all parts of the image. There is enough latitude in the beam to allow many optimal labelings to be stored. It is only after the entire image has been examined

that a single path is selected from the beam. The final labeling is therefore "the best of the best". It is found without time-consuming backtracking and it is highly accurate because it delays decision making until the entire beam has been examined.

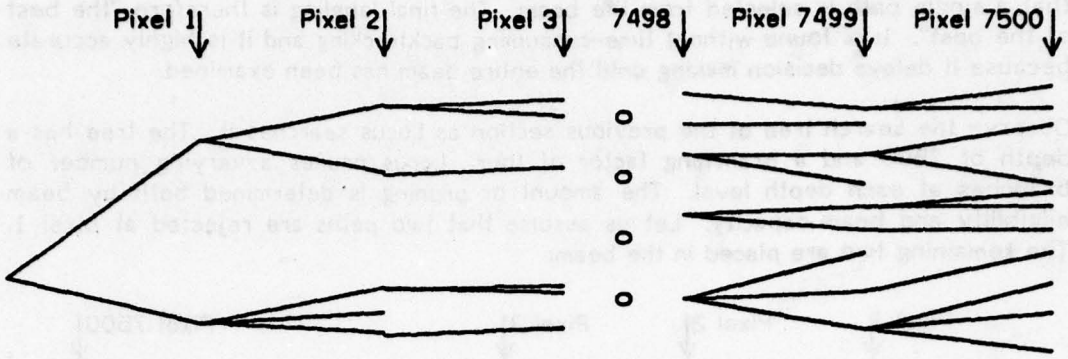
Observe the search tree of the previous section as Locus searches it. The tree has a depth of 7500 and a branching factor of four. Locus prunes a varying number of branches at each depth level. The amount of pruning is determined both by beam eligibility and beam capacity. Let us assume that two paths are rejected at pixel 1. The remaining two are placed in the beam:



There are now 8 possibilities at pixel 2. Five are selected for the beam:



Note that it is relatively easy to include many paths in the beam at this early point in the tree. However, as the tree expands, it will be necessary to make severe reductions in the percentage of paths that are saved. When the entire image has been scanned and the beam has been selected, the tree might look like this:



This completes what is known as the forward pass of the search. No unique path selections have been made, yet the entire beam has been built. All that is left is to assemble a final labeling. This step, known as the backtrace, is simply a backwards scan of the paths that were placed in the beam by the forward pass. The backtrace bears no relation to backtracking: its function is to proceed directly through the beam in reverse order without recursion, iteration, search, or any other extended computation. The backtrace is not unlike the painter who, after meticulously laboring on a painting, steps back to observe her work. The forward pass is the painter's effort; the backtrace is her final evaluation.

2.4.3: The Forward Pass

The forward pass of Locus search proceeds in breadth order through the knowledge network creating a beam-like search tree as it goes. Each level of depth in the tree corresponds to a pixel in the sensed image and each knowledge network PPE at a depth level is a possible label assignment for that pixel. The following sub-sections discuss the various aspects of the forward pass.

2.4.3.1: Order of Search

The order in which the image pixels are scanned is raster (left-to-right, top-to-bottom). For the 75x100 pixel images, this means that depth level 1 of the search tree corresponds to the pixel in the upper-left-hand corner, depth level 100 is the upper-right-hand corner, depth level 7401 is the lower left-hand corner, etc. For pre-segmented images, the segments are ordered by raster according to the position of the segment centroid[3]. This order of search may appear at first to be detrimental to the quality of labeling. It will be shown, however, that when using Locus, the order of search is of minor importance as long as the backtrace follows the reverse order of the forward pass.

3 Some re-ordering may be required to ensure that each segment adjoins previous segments in the search tree.

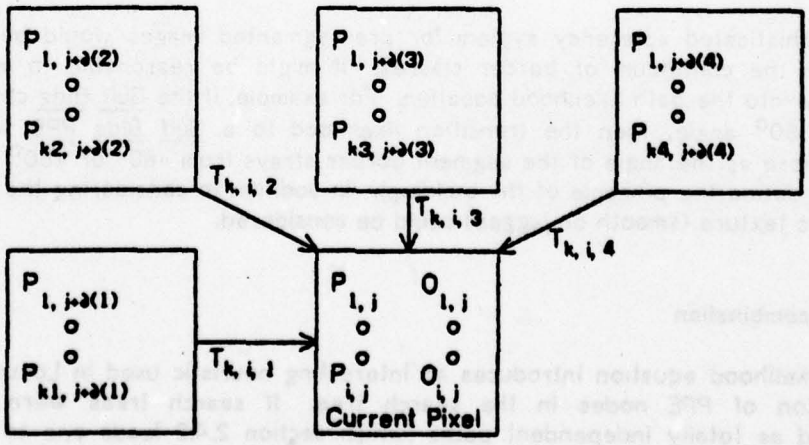
Recall that each knowledge network has an initial PPE and a terminal PPE. For the two-dimensional image task, there are actually four knowledge network PPEs which cover the initial position. These PPEs are the four image edges: top, bottom, left, and right. Locus knowledge networks can use these PPEs to help constrain region placement within the image (for example, sky is at the top of the image). Note, however, that these constraining PPEs can be used to *not* constrain simply because Locus imposes no fixed rules on region placement within the scene: it leaves that as an option.

2.4.3.2: Path Likelihoods

During the forward pass, determination of likely PPEs is based on the computation of a path likelihood for the search tree path that arrives at that PPE. The path likelihood is defined recursively in terms of the path likelihoods of all parent PPEs in the search tree[4]. It is this path likelihood value which is used to build the search tree and guide the forward pass. It uses three pieces of information: the optical match of the pixel to the PPE; the path likelihoods of previous PPEs; and the transition likelihoods of arriving from those previous PPEs. Formally:

$$P_{i,j} = O_{i,j} \times \underset{d}{\text{AVERAGE}} \left[\underset{k}{\text{MAX}} \left(P_{k,j+\partial(d)} \times T_{k,i,d} \right) \right]$$

where $P_{i,j}$ is the path likelihood of PPE i for depth level j (position j of the signal); $O_{i,j}$ is the optical match of PPE i to the pixel at position j; $\partial(d)$ is the depth adjacency *function which offsets the current search tree depth (j) to a "previous" adjacent depth* ($j+\partial(d)$) in the two-dimensional direction d; and $T_{k,i,d}$ is the transition likelihood of traveling from PPE k to PPE i in direction d (explained further in sub-section 2.4.3.4). The following diagram illustrates the physical placement of these values in an unsegmented image:



4 The recursive nature of the path likelihood equation explains why Locus needs edge (or initial) PPEs in every knowledge network.

The need for the ∂ function is explained by the fact that the task is image interpretation which is two-dimensional. Since depth within the search tree is one-dimensional, some concept of pixel adjacency must be incorporated. This is done with the ∂ function which determines, for a given two-dimensional direction, what two depth levels are physically adjacent in the image. For example, in a 75x100 image, ∂(above)=-100 and ∂(left)=-1. This is because of the raster order of scan: a pixel to the left is 1 back in the search tree whereas a pixel to the top is 100 back on the previous raster line. Pre-segmented images have an arbitrary number of adjacencies so the ∂ function can extend as far back as the start of the image.

In unsegmented images, ARGOS uses four primary directions of adjacency: left, above, upper-left, and upper-right. These four directions, which are the range of d, have four opposite directions which together allow all horizontal, vertical, and diagonal relationships to be defined within the 3x3 matrix of pixels surrounding the current point. Note that the opposite directions are not explicitly used because, for example, the relationship "A below B" can easily be expressed as "B above A".

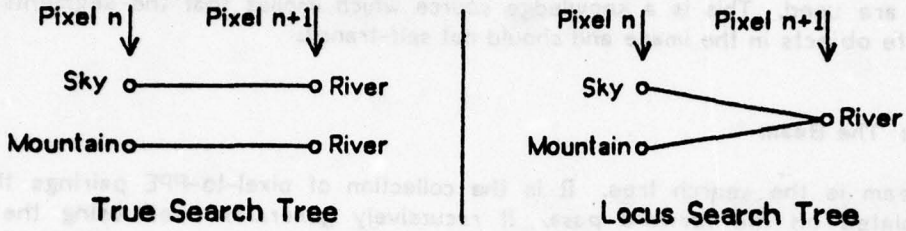
In pre-segmented images, adjacency is much different. The directions of adjacency cannot be limited to an upper-left semi-circle since the previous segment can border on all sides of the current segment. Thus, at least eight directions of adjacency are required to retain the same amount of knowledge. However, the pre-segmentation version of ARGOS does not use diagonal directions of adjacency. This is because segments which are diagonally adjacent can also be considered to adjoin horizontally and/or vertically. The result is that pre-segmentation ARGOS has the following four directions of adjacency: left, above, right, and below. These directions are used to match the segment adjacency to the network adjacency when determining the transition likelihood. Unlike unsegmented ARGOS, these directions of adjacency can be combined when describing segment adjacency. Thus two segments can adjoin in any of 16 directions (left, right, above, below, above and left, above and right, etc). Even complex relationships like "containment" (Levine, 1977) can be incorporated in this scheme.

A more sophisticated adjacency system for pre-segmented images would be able to account for the continuum of border classes. It might be reasonable to work the border type into the path likelihood equation. For example, if the Gulf Bldg comes to a point at a 60° angle, then the transition likelihood to a Gulf Bldg PPE would be penalized more as the angle of the segment border strays from -60° or +60° (the two angles that define the pinnacle of the building). In addition to considering the angle of a border, its texture (smooth or jagged) could be considered.

2.4.3.3: Recombination

The path likelihood equation introduces an interesting heuristic used in Locus search: recombination of PPE nodes in the search tree. If search trees were actually represented as totally independent paths (which section 2.4.2 leads one to believe) then any PPE-pixel node would uniquely identify its lineage back to the root of the tree. In the image understanding world, where search trees are phenomenally large, these independent paths are too expensive to retain. Therefore all paths at a given

depth level which transit to the same PPE in the knowledge network are merged by Locus into one PPE node in the search tree. For example, if both sky and mountain PPEs can transit to the river PPE, then the left-hand figure below shows the true search tree for this transition and the right-hand figure shows the Locus search tree which combines the river PPEs.



It is for this reason that the "maximum" factor exists in the path likelihood equation: the best previous PPE is saved, along with its path likelihood; all others are rejected. In a true search tree, the "maximum" factor is unnecessary because there is only one parent PPE for each child. Locus's rejection of less likely paths means that, in actuality, one of the transitions in the above Locus search tree will be ignored after both have been calculated. The rejection of a path is equivalent to the rejection of the entire path leading up to this point from the beginning of the search tree. However, the rejection is not damaging since there is now a better path to this point.

There is one more detail that should be mentioned about the path likelihood equation: the "average" clause. This clause indicates that path likelihoods from all directions of adjacency are being considered. The likelihoods are averaged to show that they contribute equally to the overall path likelihood. As a side constraint, the unsegmented version of ARGOS insists that a parent exist in all four directions. For example, if three out of four pixel neighbors have beam entries that can legally transit to the river PPE, but there are no entries in the fourth pixel neighbor that allow this, then the river PPE will not be placed in the beam at this pixel. The pre-segmentation version of ARGOS relaxes this constraint and simply penalizes any PPE that does not have transitions from all neighbors. This allowance is due to the potentially large number of neighbors that a segment can have.

2.4.3.4: Transition Likelihoods

More reduction in the complexity of the path likelihood equation can be obtained by selecting the transition likelihoods from one of two values: 0 or 1. A value of 0 indicates that it is not feasible to transit from PPE k to PPE i in direction d. A value of 1 means that the transition is allowed (i.e. PPE k and PPE i may adjoin each other in the d direction). The transition likelihoods are therefore the knowledge network constraints and comprise the major knowledge element of the path likelihood equation.

ARGOS actually implements the transition likelihoods as one of three values: 0 for a disallowed transition; 0.1 for an allowable transition from one PPE to another; and 0.9

for an allowable transition from a PPE to itself. Self-transitioning is a necessary special case in the unsegmented system since most PPEs cover a large area of pixels. The increased likelihood of transitioning to the same PPE is a form of knowledge which ARGOS uses to label city scenes. The value of 0.9 is taken, without experimental verification, from the Harpy speech implementation of Locus. Pre-segmentation ARGOS also uses differing intra-state and inter-state transition likelihoods, but the reverse values are used. This is a knowledge source which implies that the segments form complete objects in the image and should not self-transit.

2.4.3.5: The Beam

The beam is the search tree. It is the collection of pixel-to-PPE pairings that is accumulated on the forward pass. It recursively generates itself using the path likelihood equation to select new members. However, not all of the path likelihoods produced by the equation are saved in the beam. This is because Locus resembles a first-order Markov system, so the only likelihood values that are needed are for the à calculations in the immediate neighborhood of the pixel being computed. In the unsegmented system, anything more than 1 scan-line back (100 depth levels back) can be discarded because path likelihoods exist only to compute other path likelihoods. The pre-segmentation system, however, must keep all of the likelihoods since strange segment adjacencies may require arbitrary values from the beam.

The important information in the beam is the PPE connections. Whenever a path likelihood is computed, there is an "optimal parent" in each direction of adjacency. This optimal parent is considered to be a major contributor to the path likelihood at the current pixel or depth level. It can be seen in the path likelihood equation as the *maximum* k for each direction d. The collection of optimal parents is what makes up the beam.

2.4.3.6: Pruning

Yet another distinguishing feature of Locus is its pruning. It has been mentioned that Locus selects the best few PPE connections and saves them in the beam. There is a fixed maximum size at each level of the beam which is considerably smaller than the number of PPEs. Therefore, only those PPEs with the highest likelihoods are allowed in the beam. Those that cannot fit are usually not important in the overall labeling scheme because there are many other beam entries with higher likelihoods.

In addition to fixing the beam size, Locus maintains a threshold below which a likelihood will be pruned regardless of beam space availability. This threshold, which is used in Harpy Locus, is dynamic in that it is relative to the best likelihood at the particular depth level. If the best PPE has likelihood value P, then all other PPEs with likelihoods below P-threshold will be pruned. It has been found that a threshold of 100 (out of a possible range of 256, see next subsection) will prune on the order of 60% of the PPEs at each depth level. This is a hefty cut yet it does not damage the labeling quality.

2.4.3.7: Normalization

One problem that arises when computing the path likelihoods is precision. In a true Markov system, all of the probabilities[5] at a given depth level must sum to 1. In Locus, however, there is much pruning and re-combination of likelihoods. The result is that the values degenerate as the network is scanned. To prevent this, the likelihood values at a pixel are normalized once they have all been computed. The most promising PPE at the level is given the likelihood value 1 and all others are linearly scaled to fall below that. This normalization causes no damage to the algorithm because Locus is only concerned with one single depth level as it relates to another. If all PPEs at a level are normalized together, then the relative results are the same.

As an added bonus, the normalization allows the path likelihoods to be stored with very few bits of precision. ARGOS uses negative log values of all likelihoods so that it can add instead of multiplying. Also, it is able to store these as integral values in only 8 bits of precision. Thus, the likelihood value 1.0 is stored as a 0 and the likelihood value 0.0 is stored as 255. All fractions between 0.0 and 1.0 are represented as an integer between 0 and 255.

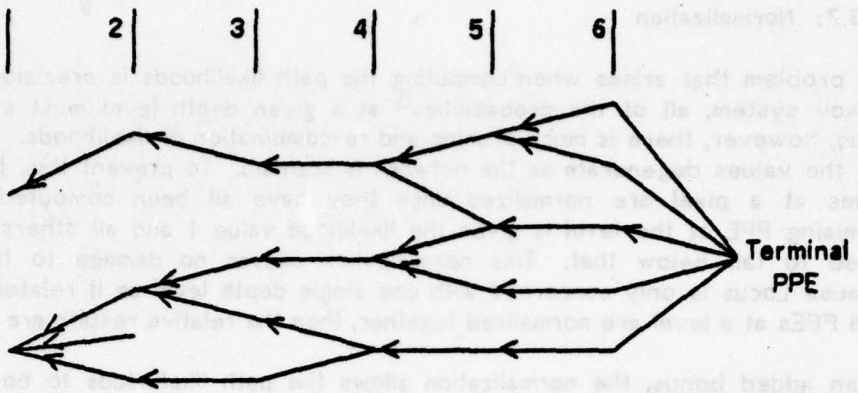
2.4.4: The Backtrace

After the entire image has been scanned, all that remains is the beam: stretching out for as many depth levels as there are pixels in the image, and containing PPE connections from the forward pass. The backtrace scans the beam in reverse order (i.e. from the bottom of the search tree to the top) and produces a labeling of the image. The labeling, of course, consists of a series of PPE assignments to every pixel in the image.

The inquisitive reader will ask why the beam is scanned in reverse order. The answer is: to obtain a unique labeling. Notice that the beam contains, for each PPE entry, a pointer to its optimal parent PPE. This optimal parent pointer is a direct by-product of the path likelihood equation. Therefore, when scanning the beam backwards, each child PPE which is selected will identify exactly one parent PPE at the next higher level. If the beam were scanned forwards, there would be ambiguity when a parent had more than one child: which child should be selected next? Observe the following beam:

---

5 I use the term "probability" only when discussing true statistical systems. Locus is pseudo-statistical, so I remove any theoretical implications that may be associated with the numbers by calling them "likelihoods".



There are six depth levels, plus the terminal PPE[6]. Notice that the beam has exactly one path defined by following the optimal parent pointers (arrows) from the terminal PPE. The same cannot be said for the other order. The backtrace is necessary in this form of search tree evaluation because of the re-combination that occurs when two PPEs at a depth level join into a third at the next depth level. If there were no re-combination, then the optimal PPE at the bottom depth level would uniquely define a path back through the search tree without the need for a beam or any parent pointers. However, the combinatorics of search without re-combination are prohibitive in the image understanding task, so Locus re-combines and does a backtrace.

Notice that the backtrace is extremely fast. It performs no search or other involved computation. In fact, the backtrace is linearly bounded in time to the size of the image. Even with the complications that are about to be discussed, it retains these qualities.

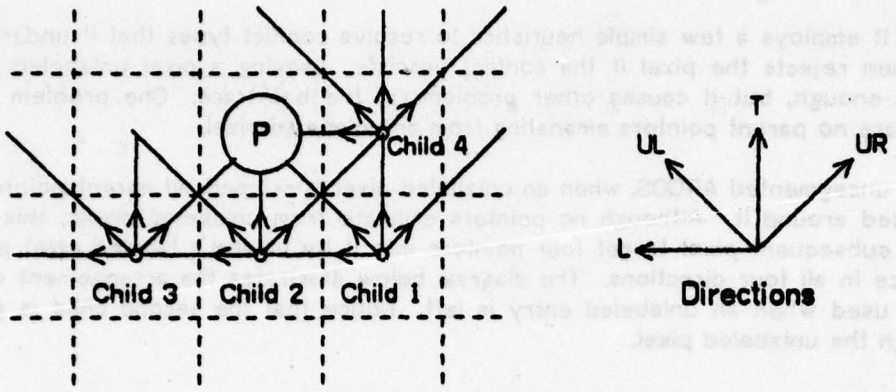
2.4.4.1: Conflicts: The Problem

Conflicts arise in the backtrace when multiple beam pointers from the forward pass disagree. This is a subtle problem that is once again caused by topology: the task is two-dimensional but the beam is one-dimensional. The solution to the dimensionality problem is simple but it introduces problems in the backtrace.

To handle two-dimensional data on the forward pass, there is an optimal parent in each of the directions of adjacency. This was mentioned in section 2.4.3.2 when the path likelihood equation was presented. If the forward pass saves d parents for every child then, by symmetry, there must be d child PPEs for every parent on the backtrace. It looks like this in the unsegmented system where d=1:

---

6 The terminal PPE is defined to be that PPE to which all those at the last depth level must transit. Therefore, all of the paths in the beam end up at this single point. The terminal PPE exists only as a formalism in Locus: without it, the bottom-level PPEs would have to be evaluated to see which is the best one for starting the backtrace. By using the concept of a terminal PPE, the best bottom-level PPE "falls out" as an optimal parent.



In the above diagram, the backtrace is about to label pixel P and the four child PPEs have already been scanned (remember: the backtrace proceeds in reverse raster order, from right-to-left, bottom-to-top). Each child pixel has been assigned a PPE label, therefore each child pixel has something to say about its parents in four directions (observe the tangle of arrows). Notice that although there is only one optimal parent for every child, there are four directions of adjacency, so there is an optimal parent in each direction. Parent pointers connect two levels of the search tree and every PPE has parents at four other levels, therefore four parent pointers. If, for each child pixel, c ($1 \leq c \leq 4$), there are four parents $OP_c[d]$ (d taken from direction key above), then the four PPE label choices for pixel P are OP_1[UL], OP_2[T], OP_3[UR], and OP_4[L].

Clearly, two-dimensionality presents a problem of conflicts: when the children select different parent labels. Although it would be nice to have unanimous PPE selection, this is not always the case. Conflict resolution is an interesting problem that has not been fully explored. Therefore, the following discussion contains some techniques and observations, but no definitive solution.

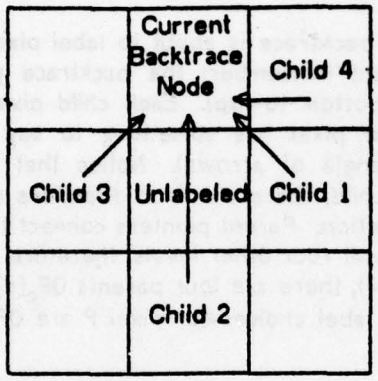
2.4.4.2: Conflicts: Some Answers

There are two ways to resolve conflicts. The obvious one is to select one of the candidates from the child PPEs through some heuristic. The surreptitious solution is to throw out a pixel when it has a conflict. This latter technique is not as bad as it sounds for a few reasons. First, there is no rule which says that every pixel must be labeled. Humans don't assign interpretations to every point in an image, so why must ARGOS? In effect, the system can say, "this point is confusing: it could be any of these objects, I can't tell. I'll leave it undecided." The second reason for leaving conflict points unlabeled is in direct support of the first: experience has shown that conflicts arise only at region borders or in areas where there is inadequate training. When ARGOS changes PPEs, it gets confused for a pixel or two but soon picks up the scent and continues faithfully.

Given that conflicts can be resolved by arbitration or rejection, ARGOS chooses to do

both. It employs a few simple heuristics to resolve conflict types that it understands, and then rejects the pixel if the conflict persists. Leaving a pixel unlabeled sounds simple enough, but it causes other problems in the backtrace. One problem is that there are no parent pointers emanating from an unlabeled pixel.

In the unsegmented ARGOS, when an unlabeled pixel is skipped, all parent pointers are extended around it. Although no pointers emanate from unlabeled pixels, this allows every subsequent pixel to get four pointers into it by having a labeled pixel at some distance in all four directions. The diagram below illustrates the arrangement of child pixels used when an unlabeled entry is left. Notice that the second child is passing through the unlabeled pixel.



In pre-segmentation ARGOS, unlabeled segments do not affect the backtrace: they are simply ignored. Since the number of child segments varies anyway, it does not matter if there are one or two less children available when it comes time to label a segment. It should not be presumed, however, that unlabeled segments are harmless. Conflicts indicate inconsistency in the interpretation and should be resolved whenever intelligently possible.

The most powerful, and least obvious conflict resolution technique is simple knowledge constraint checking. The child pixels have been labeled already so their parent PPE selections should be consistent with all legal adjacency rules in the relational knowledge network. It is simple enough to check each candidate for the parent position against its children and reject those candidates that are inconsistent. If all but one candidate are rejected, then the conflict has been resolved. If there are still multiple candidates, or, if all of the candidates are rejected, then some other resolution technique must be employed.

Before moving on to other resolution strategies, it is worthwhile to stop and see why the above technique is able to function. After all, aren't all beam entries generated from the path likelihood equation? And doesn't that equation use only legal PPE relationships? So how can invalid relationships arise? The answer has to do with the reverse order of scanning and the topology. $\underline{x}$ may adjoin $\underline{y}$, and $\underline{y}$ may adjoin $\underline{z}$, but $\underline{x}$ doesn't have to adjoin $\underline{z}$. Even though they may end up touching in two-dimensions, they do not have to be legally adjacent. Don't forget that the beam contains many

likely label candidates which may not all be consistent with each other. For a good demonstration of how this type of conflict can arise, see the example in section 2.5.

The next technique used in conflict resolution is voting. It has been found that if one candidate outnumbers all the others, then that candidate should be chosen for the pixel label. Voting is easily implemented and aids in labeling quality.

The only other resolution technique that was explored is directional preference. This last-resort heuristic is used only when the above two techniques fail. What it does is to select candidates solely on the basis of their adjacency direction. The assumption is that certain directions aid the labeling more strongly than others. When this assumption was tested out in the unsegmented system, it was found that the diagonal directions seem to be slightly stronger than the horizontal and vertical directions. However, the advantage is not strong enough to be significant. In addition, use of this technique implies that all conflicts can be resolved since this technique cannot fail. Inaccuracies in labeling rise faster than accuracies when conflict points are forced to be labeled. Therefore, directional preference is not used.

There are a number of conflict resolution techniques that were not explored. An example of one of these is the use of optical matches. This factor can be used in conjunction with other techniques (i.e. as a weighting factor for voting) or by itself. In fact, any knowledge that is used in the forward pass can also be applied in the backtrace to help resolve conflicts.

2.4.5: Advantages of Locus Search

Locus performs very well in labeling images. The technique of delaying decision until the backtrace pass allows a globally near-optimal labeling to be selected. "Globally near-optimal" means that every knowledge constraint which has not been pruned is able to be applied to every PPE relation in the scene. For example, the discovery of a boat in the lower-left corner can affect the selection of a boat house in the upper-right corner or anywhere else. Of course, pruning does allow for the possibility of missing the globally optimal path. However, this happens too seldom to make the added cost of exploring all nodes worthwhile.

The reason that Locus search yields globally near-optimal results is simple: it resembles a Markov system, and Markov systems yield globally optimal results. In fact, if no pruning, normalization, or node recombination were done, then Locus *would* be a Markov system. It has been found, through extensive experience with Locus systems, that these heuristics do little damage to the Markov nature of the search. Thus the quality is preserved.

Yet another advantage of Locus search is reduced order dependence. It was mentioned earlier that the raster order of search is not detrimental to labeling quality. In a Markov system, all that matters is that every point be examined: they are all

related to each other regardless of order. In Locus, all that matters is that every point be examined and that the backtrace reverse the order of the forward pass. Given that Locus implements a Markov system (which, within heuristic bounds, it does) then it becomes marginally important *how* the image is scanned. In ARGOS, the raster scanning order is done for two reasons: to provide a consistently high number of neighbors in the context of the search (i.e. every pixel in the forward pass or backtrace is guaranteed to have neighbors that have already been examined) and to demonstrate that the order of search is not significant.

It would be feasible to add search order as a knowledge source. This would require the addition of a depth level factor in the path likelihood equation (perhaps somewhere in the normalization phase). Then the search could start from one or many "points of interest" in the sensed image and proceed outward (perhaps in a spiral). These points of interest could be selected on the the basis of their signal features (i.e. bright areas) or on the basis of some knowledge about the image (perhaps from a higher knowledge hierarchy level).

Another advantage of Locus search is speed. Locus uses no backtracking or other unbounded computation. The beam constrains all search possibilities to a reasonable size as it follows many parallel paths through the search tree. In fact, the varying sized beam can be functionally compared with the varying sized stack in standard backtracking searches: both list the current best paths through the search tree. The difference is that the beam is horizontal and the stack is vertical (when search trees are viewed as top-to-bottom, not left-to-right). It is this directionality which allows Locus to limit the beam size without damaging the search.

2.5: AN EXAMPLE

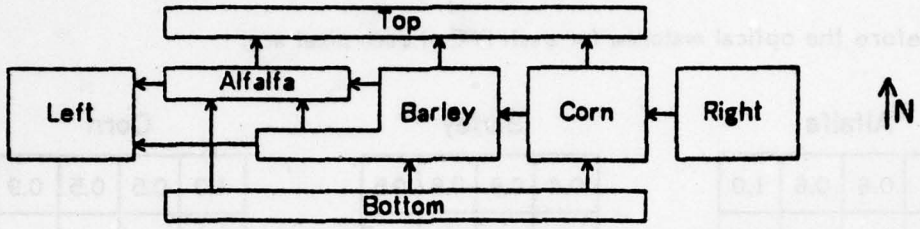
A good way to finish up this chapter is with an example. This example is, of necessity, a toy use of Locus but it will make use of much that has been presented in this chapter to show how it all fits together. Those readers who are satisfied with the details of Locus may skip to the next chapter.

2.5.1: Background

This example concerns the labeling of a very small satellite picture of a fictitious agrarian nation. The nation's chief products are Alfalfa, Barley, and Corn (sometimes refered to in this example as A, B, and C). Due to soil conditions, winds, and antiquated laws, farmers always plant their crops in fixed positions on their land. The law states that Barley and Corn must be planted, but Alfalfa is optional. The soil conditions

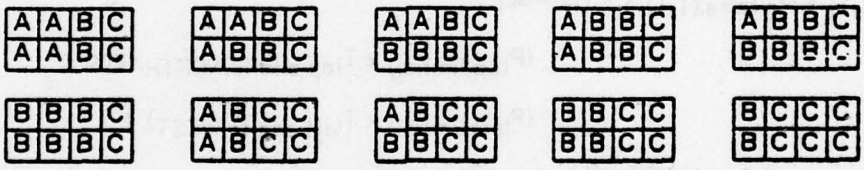
mandate that Corn always be planted to the east of Barley, and winds force farmers who choose to plant Alfalfa to do so in the north-west corner of their field.

A Locus knowledge network is compiled from the above information. It uses only north and west in its relationships and looks like this:



Needless to say, there are three PPEs: alfalfa, barley, and corn. Together with the four image-edge PPEs, they define the knowledge network. The goal is to label the satellite photographs and identify the crop types.

The satellite photographs are not very large: four pixels across and two pixels high (unsegmented). However, with a total of 8 pixels and 3 choices of a label for each pixel, there are still over 6000 possible labelings of each satellite photograph. It was mentioned earlier that knowledge does most of the reduction of the search space and this example is no exception. In fact, the knowledge-reduced search space actually contains only the following ten possible labelings:



2.5.2: Low Level

The low-level side of this example is as follows: there is one sensor in the satellite which generates brightness values in the range of 1 to 10. Therefore, the feature vector templates are simply one number which describes each PPE. Let us assume that the typical alfalfa field registers 4 on this sensor, barley registers 9, and corn registers 3.

For a distance metric, it is adequate to choose an unweighted subtraction measure. In particular, the formula $O_{i,j} = 1 - |i-j|/10$ will yield a fraction between 0 and 1 which indicates the likelihood that PPE i is matched to image pixel j. In our example, the sample photograph to be labeled looks like this:

(1, 1)	(1, 2)	(1, 3)	(1, 4)
3	8	8	4
(2, 1)	(2, 2)	(2, 3)	(2, 4)
8	6	5	2

so therefore the optical matches for each PPE at each pixel are:

Alfalfa

0.9	0.6	0.6	1.0
0.6	0.8	0.9	0.8

Barley

0.4	0.9	0.9	0.5
0.9	0.7	0.6	0.3

Corn

1.0	0.5	0.5	0.9
0.5	0.7	0.8	0.9

2.5.3: The Search

The search through the image now begins. The first step is the forward pass which starts with the pixel at position (1, 1) in the upper-left corner. The path likelihood for alfalfa at pixel (1, 1) is derived as follows:

$$P_{alfalfa,(1,1)} = O_{alfalfa,(1,1)} \times \text{AVERAGE}[$$

$$(P_{top,(initial)} \times T_{top,alfalfa,NORTH}) \text{ and}$$

$$(P_{left,(initial)} \times T_{left,alfalfa,WEST})]$$

$$= 0.9 \times \text{AVERAGE}[(1.0 \times 1.0) \text{ and } (1.0 \times 1.0)]$$

$$= 0.9$$

Notice that the initial PPEs are defined to have the likelihood value 1.0 and that all transition likelihoods are simply 1 or 0 depending on the PPE connection. By similar computation, $P_{barley,(1,1)} = 0.4$ and $P_{corn,(1,1)}$ is undefined. This last path likelihood is undefined because the transitions are not allowed in the WEST direction. Once both valid PPEs have been calculated for pixel (1, 1), the beam is updated. In this case, barley is pruned because its path likelihood is too low. (In this example, anything .5 or more from the best likelihood is pruned. Barley has the value 0.4.) The remaining value is normalized to 1.0.

At pixel (1,2), the three PPEs are evaluated again. Observe the path likelihood of barley at pixel (1, 2):

$$P_{barley,(1,2)} = O_{barley,(1,2)} \times \text{AVERAGE}[$$

$$(P_{top,(initial)} \times T_{top,barley,NORTH}) \text{ and}$$

$$(P_{alfalfa,(1,1)} \times T_{alfalfa,barley,WEST})]$$

$$= 0.9 \times \text{AVERAGE}[\ (1.0 \times 1.0) \text{ and } (1.0 \times 1.0)\ ]$$

$$= 0.9$$

At pixel (1, 2), corn is again pruned because there is no barley to the west of it. Therefore, only two PPEs are valid, neither of which is pruned. At pixel (1, 3), all three PPEs are retained in the beam. This is the computation of barley at (1, 3):

$$P_{barley,(1,3)} = O_{barley,(1,3)} \times \text{AVERAGE}\{$$

$$(P_{top,(initial)} \times T_{top,barley,NORTH}) \text{ and}$$

$$\text{Max}[\ (P_{alfalfa,(1,2)} \times T_{alfalfa,barley,WEST}),$$

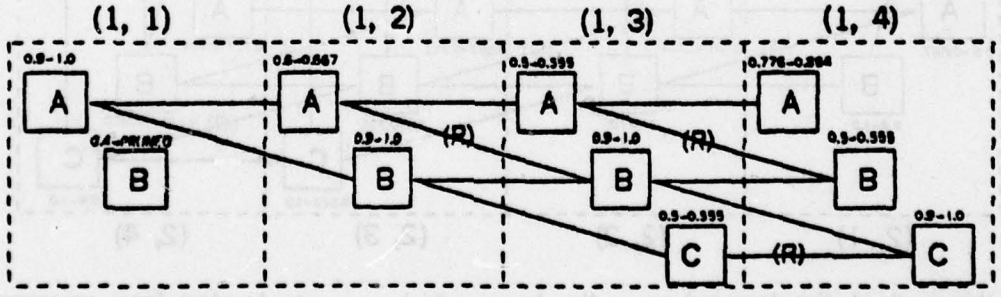
$$(P_{barley,(1,2)} \times T_{barley,barley,WEST})\ ]\ \}$$

$$= 0.9 \times \text{AVERAGE}\{\ (1.0 \times 1.0) \text{ and } \text{Max}[\ (.667 \times 1.0), (1.0 \times 1.0)\ ]\ \}$$

$$= 0.9 \times \text{AVERAGE}\ \{\ 1.0 \text{ and } 1.0\ \}$$

$$= 0.9$$

The diagram below shows the status of the search tree after the top row of the image has been scanned:



The two numbers above each PPE node in the search tree indicate the path likelihood values before and after normalization. Observe that certain paths are rejected (R) because there are more optimal parents. The forward pass now advances to the second row and scans it from left to right (west to east). Pixel (2, 1) has only two choices: alfalfa (normalized likelihood 0.667) and barley (likelihood 1.0). Pixel (2, 2) also has these two possibilities because corn was not allowed at pixel (1, 2). Lets examine the likelihood calculation for barley at pixel (2, 2). It is quite complex:

$$P_{barley,(2,2)} = O_{barley,(2,2)} \times \text{AVERAGE}\{$$

$$\text{Max}[\ (P_{alfalfa,(1,2)} \times T_{alfalfa,barley,NORTH}),$$

$$(P_{barley,(1,2)} \times T_{barley,barley,NORTH})\ ] \text{ and}$$

$$\text{Max}[\ (P_{alfalfa,(2,1)} \times T_{alfalfa,barley,WEST}),$$

$$(P_{barley,(2,1)} \times T_{barley,barley,WEST})\ ]\ \}$$

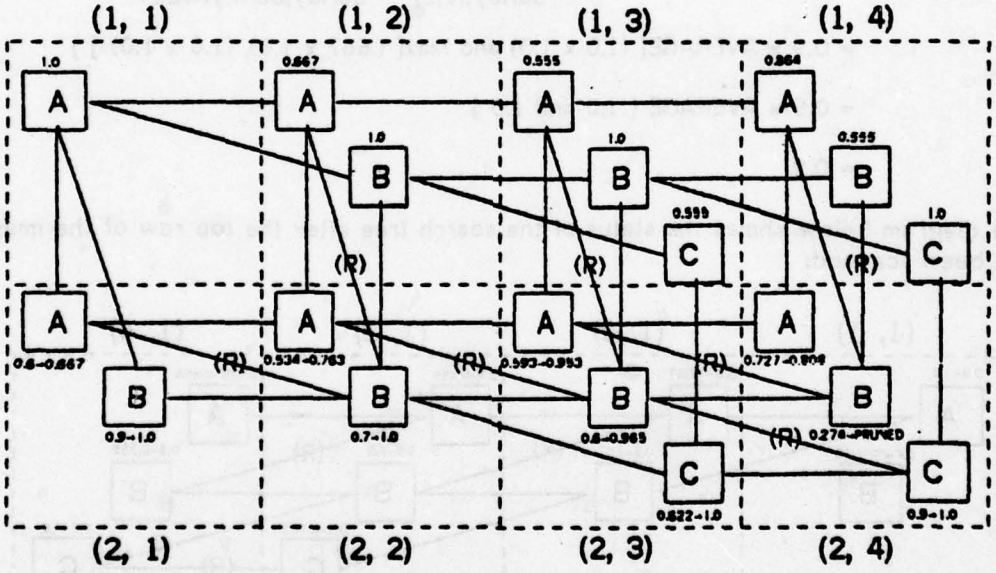
$$= 0.7 \times \text{AVERAGE}\{\ \text{Max}[\ (0.667 \times 1.0), (1.0 \times 1.0)\ ] \text{ and}$$

$$\text{Max}[\ (0.667 \times 1.0), (1.0 \times 1.0)\ ]\ \}$$

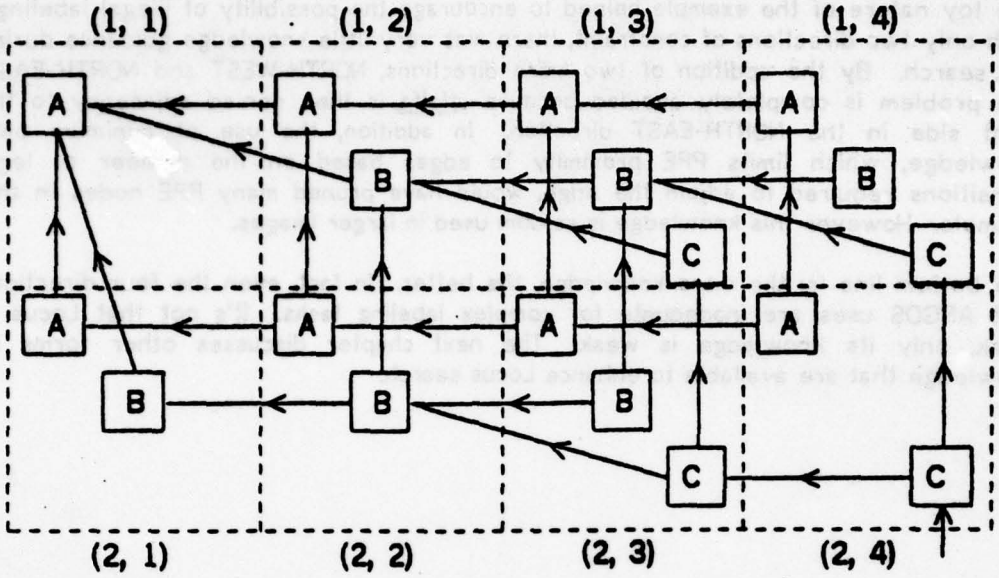
$$= 0.7 \times \text{AVERAGE}[\ 1.0 \text{ and } 1.0\ ]$$

$$= 0.7$$

The completed forward pass, with row two, is shown below. Notice that row one has been simplified to its pruned normalized paths.



At the end of the forward pass, the beam, which was constructed from unpruned optimal paths, looks like this:



The backtrace starts at pixel (2, 4). The terminal PPE (arrow entering from the bottom) indicates the corn label for this pixel because it had the highest path likelihood on the forward pass. From there, the corn at (2, 3) is selected (just follow the arrows) and barley is selected at (2, 2) and (2, 1). Next, pixel (1, 4) is labeled and corn is easily picked. At pixel (1, 3), however, there is a conflict. The child at (2, 3) recommends corn but the child at (1, 4) recommends barley. The conflict is resolved by observing that the barley candidate would be inconsistent with the corn to the south of it, whereas the corn candidate is consistent with both children. There are no other conflicts in the backtrace and the final two pixels are quickly and unanimously selected to be barley and alfalfa. The final labeling is:

Alfalfa	Barley	Corn	Corn
Barley	Barley	Corn	Corn

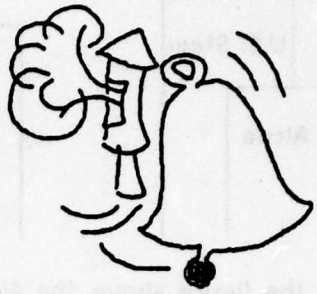
2.5.4: Comments

A dangerous situation arose in the above labeling which could have had disastrous results. The pixel at (2, 4) was correctly labeled corn but could have been labeled alfalfa. If alfalfa had been stronger, the entire scene would have been labeled alfalfa and that is an illegal labeling. How could that happen?

The toy nature of the example helped to encourage the possibility of illegal labelings. With only two directions of constraint, there was very little knowledge guidance during the search. By the addition of two more directions, NORTH-WEST and NORTH-EAST, this problem is completely avoided because alfalfa is then denied adjacency to the right side in the NORTH-EAST direction. In addition, the use of minimum path knowledge, which limits PPE proximity to edges based on the number of legal transitions required to adjoin the edge, would have pruned many PPE nodes in this example. However this knowledge is seldom used in larger images.

The bottom line is: the more knowledge, the better. In fact, even the four directions that ARGOS uses are inadequate for complex labeling tasks. It's not that Locus is weak, only its knowledge is weak. The next chapter discusses other forms of knowledge that are available to enhance Locus search.

CHAPTER 3: KNOWLEDGE



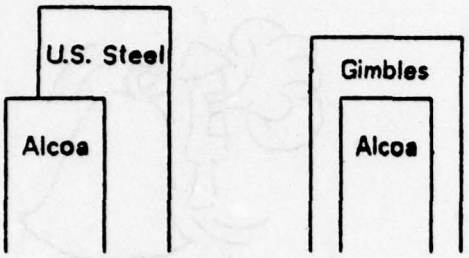
Locus search, as the last chapter presented it, uses only one kind of knowledge: region adjacencies. This adjacency knowledge forms the basic structure of the network that Locus searches, but it is not the only knowledge that can be brought to bear. In fact, as this chapter will show, there appears to be no major limitation to the knowledge that can be used in Locus search.

3.1: ADJACENCY KNOWLEDGE

Before examining additional forms of knowledge, it is appropriate to discuss the exact method used to acquire region adjacency knowledge. Locus networks contain the extracted adjacency knowledge from many hypothesized views of an internal model of the scene. The internal model, which in the case of Pittsburgh city scenes is constructed from street maps, is a three-dimensional model of the city which can be used to generate all possible views of the city. To build a knowledge network, selected views (or hypotheses) are generated and added as separate paths[1].

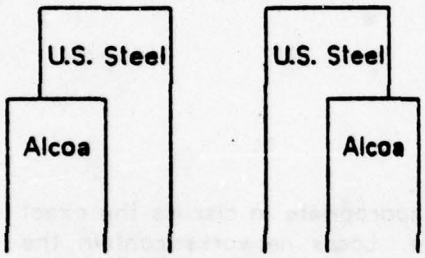
The non-obvious aspect of building a network from multiple views is that a region which appears on different views is sometimes represented with separate PPEs and at other times is merged into one PPE. The decision to merge is based on a similarity measure between the PPEs. Observe the following two views of the Alcoa Building:

1 See Chapter 4 on knowledge hierarchies for a discussion of how the views are selected.



Alcoa1	below	U.S. Steel
Alcoa1	leftof	U.S. Steel
Alcoa2	below	Gimbles
Alcoa2	leftof	Gimbles
Alcoa2	rightof	Gimbles

In the figure above, the Alcoa Bldg appears in front of the U.S. Steel Bldg in one view and in front of Gimbels Dept. Store in another view. It is represented with two PPEs in the knowledge network that reflect the five adjacencies listed above. This is because the two views happen to be from opposite directions and there are no common adjacencies between the two Alcoa Bldg regions. If, however, two views show the Alcoa Bldg alternately to the left of the U.S. Steel Bldg and to the right of the U.S. Steel Bldg as in the following diagram, then these two instances of the Alcoa Bldg are likely to be merged into one PPE which has the transitions shown.



Alcoa	below	U.S. Steel
Alcoa	leftof	U.S. Steel
Alcoa	rightof	U.S. Steel

The decision to merge PPEs from different views is based on many factors including the relative size and position of the two PPE regions. For a detailed discussion of the merging algorithm, see section 5.2.1 on network size reduction.

Although some knowledge is necessarily lost when regions are merged into one PPE, it is mandatory that the networks be reduced or else each hypothesized view will be stored as a separate path through the network. This not only makes the networks too large to search, but forces the beam to contain at least one path for every view, if that view is to be considered in the labeling. It has been found that frequent region merging produces optimal labeling results because little information is lost and the search is much easier (see section 5.2.2).

The remainder of this chapter discusses how the location, size, and shape of a region can be used to aid the Locus labeling process. Even pre-segmentation, where arbitrarily shaped areas are presented for labeling, can be considered to be a source of knowledge.

3.2: PRE-SEGMENTATION

ARGOS does not explicitly segment: it labels. Segmentation is the division of an image into distinct areas, each of which is a separate object in the image. Labeling is the identification of each object. Many image understanding systems segment their image first and then label the segments (Sakai et. al., 1976; Williams et. al., 1977). ARGOS is able to take pre-segmented images and label the segments, but it is also able to take unsegmented images and "post-segment" them. What this means is that it is possible to view the labeled output as a segmentation since the labels are grouped into distinct clusters within the image.

Pre-segmentation, therefore, is the use of images that have been segmented (by some unknown but reliable algorithm) *before* ARGOS labels. Some reasonable choices of pre-segmentation algorithms are clustering (Ohlander, 1975), and other multispectral classifications (Kettig and Landgrebe, 1976; Rodd, 1972). ARGOS is currently running with a clustering algorithm that is derived from Ohlander's work (Shafer and Kanade, 1978). The use of pre-segmentation has the advantage of time and space saving because there are many fewer nodes in the search tree, typically two orders of magnitude fewer. In addition, there is increased accuracy when using pre-segmentation since smaller numbers of search tree nodes can be constrained better. The only disadvantage of pre-segmentation knowledge is that care must be taken to ensure that enough segments are found. It is only marginally harmful if there are too many segments because the labeler can simply give multiple segments the same label, but if there are too few segments, then there will be continuity gaps in the adjacency and the search will fail.

3.3: LOCATION KNOWLEDGE

Everyone knows that mountains are usually found in the top part of an image. This sort of location knowledge is separate from adjacency: it is absolute position as opposed to relative position. The use of location knowledge is quite easy and requires only two steps: learning the knowledge and using the knowledge.

Learning location knowledge is done at the same time as the learning of region adjacencies. Each hypothesized image that is combined into the knowledge network has a minimum-bounding-rectangle (MBR) drawn around each region. A minimum-bounding-rectangle is simply the rectangle formed by lines parallel to the X and Y axes which pass through the smallest X, largest X, smallest Y, and largest Y co-ordinate in the object. Thus, it is the smallest box that can be drawn around the object which is parallel to the X and Y axes. When regions from two hypothesized images are merged, their MBRs are combined into a new MBR that includes both regions. The resulting knowledge that is extracted is an MBR for each PPE. In some cases, the MBR contains useful information and in other cases it doesn't. If, for example, a building appears in

many different places across the hypothesized images, then the knowledge contained in the combined MBR will not constrain the building's location. But that is a form of knowledge which says: this building may appear anywhere.

It should be noted that the proximity of the MBRs is used as a factor when deciding whether or not to merge two regions. This prevents the creation of networks that have meaningless location information.

It would be possible to extract an exact template of the pixels occupied by each region. When regions merged, the new template would be formed from the union of the old templates. However, this much detail is not necessary for Locus since the hypothesized views are imprecise at the outset.

Using location knowledge is easier than obtaining it. In the unsegmented system, each pixel that is being evaluated during the forward pass is either in the MBR of a PPE or it isn't. If it is in the MBR, then no action is taken. If it is outside, then a penalty is added to the path likelihood equation for that PPE-pixel node in the search tree. The pre-segmentation version of ARGOS uses the segment centroid to determine location violation in the same manner. The penalty gets stronger as the pixel gets farther from the MBR of the PPE. Thus, Locus doesn't *reject a label assignment if it violates* location knowledge, but the likelihood begins to decrease as a pixel strays from its expected location. Unless there are other factors in the path likelihood equation that *override the location penalty, the path will soon be pruned from the beam.* This is how Locus search allows imprecise knowledge to be used without completely destroying the labeling.

3.4: SHAPE KNOWLEDGE

Shape is hard to define because it has many aspects. This section will discuss a number of methods of describing shape. Some are better suited to pre-segmented images, others work best with unsegmented images, and a few are totally useless to ARGOS.

When ARGOS uses pre-segmented images, it must match the shape of an area of the image to the expected shape in the knowledge network. It is not sufficient to pre-compute the shape of each segment in the image since segments may combine during the search due to self-transitions in the network. ARGOS must be able to dynamically evaluate the shape of a group of segments and compare it to the shape specified by the PPE which is labeling these segments. Although there are many choices of shape descriptors including bit masks and moment invariants (Hu, 1961), four simple shape measures were selected which make use of the segment perimeter and area.

The simplest shape measure used by pre-segmentation ARGOS is fractional fill. This is

the ratio of the area of the segment to the area of its minimum bounding rectangle. Another measure of shape is called compactness. This is the ratio of the perimeter squared to the segment area. Both of these measures distinguish compact segments from "loose" segments. Two other shape descriptors that are used are orientation and elongation. These are derived from the first moment of the Fourier transform and indicate the angle of an elongated segment and the amount of elongation. For more detail, see the thesis of Keith Price (Price, 1977).

The shape measures used by pre-segmentation ARGOS are matched with the same weighted Euclidean distance metric that was described in section 2.3.2. The only detail that must be observed during this process is the fact that orientation is cyclic, so the distance between 10^{o} and 350^{o} is 20^{o}, not 340^{o}. The results of the distance metric are used as a penalty in the path likelihood equation, thus allowing shape to collaborate with the other knowledge sources.

The above set of shape measures does not completely describe a segment's shape. The only perfect shape descriptor is a copy of the desired shape. It would not be terribly expensive to retain a template for each PPE in the knowledge network which would describe the exact shape of the expected region. Unfortunately, it is difficult to apply this kind of knowledge to Locus search because it is hard to determine *where* in the template a given search tree node is. For example, notice the following template for a cross-shaped PPE region:

		1	2		
		3	4		
5	6	7	8	9	10
11	12	13	14	15	16
		17	18		
		19	20		

This template describes a region with 20 points. If an isolated instance of that PPE is detected during the forward pass, which element of the template should that pixel be associated with: 1 or 5? It is not enough to remember all previous instances of that PPE in the beam because, after the first row of the template has been scanned, there will be uncertainty about where to place the pattern unless it appears in the beam *exactly* twice. Matching irregularly shaped segments is even harder. The multiple option philosophy of Locus indicates that some uncertainty is expected in the forward pass so it doesn't know what to do with precise knowledge. Besides, the use of extensive "look-back" in the search is contrary to the heuristics used in Locus because of the amount of time and space needed. And, of course, there is the problem of what template to store for PPEs that have been created from merged views of a region. The template method of storing shape knowledge is nice, but effective use of it in Locus search is not well understood.

It might be possible to incorporate shape templates into the backtrace. These templates would be used to help resolve conflicts by rejecting options that do not fit. It would be easier to work the precise knowledge of a template into the backtrace because there is no uncertainty: each PPE selected by the backtrace is unique to that pixel position.

A more natural shape descriptor for the forward pass (which is used in unsegmentation ARGOS) is dimensional shape: the specification of shape by the use of PPE dimensions at various angles. In the above cross example, the dimensions of the PPE region would be taken along the four directions of adjacency. The horizontal dimension would be 2-to-6 pixels, meaning that any horizontal slice through the region contains at least 2 and at most 6 pixels in a row. The vertical dimension is the same, and the two diagonals are both 1-to-4 pixels. Dimensional shape is well suited to descriptions of regular objects like buildings. However it fails in any attempt to describe complex shape like a skyline. It will be shown, however, that even a skyline can be described to Locus search.

Dimensional shape is easy to extract from the hypothesized views and, of course, is used as a factor in region mergings. Note, however, that the accumulation of dimensional shape can easily lead to useless information if the hypothesized views are taken too literally. Therefore, ARGOS rejects abnormal range values during the accumulation of dimensional shape. For example, suppose a building has one pointed turret above a wide main structure. Rejection of the abnormally narrow turret point would prevent the PPE description from having a horizontal lower dimension of 1 when there is really only one point in the PPE that is so narrow.

Dimensional shape is easy to implement in the forward pass of Locus search. All that is needed is four counters with every beam entry that specify the dimension (up to that point in the raster scan) of that PPE. Whenever a PPE is added to the beam, its dimensions are simply set to be one greater than the four surrounding PPE dimensions in the beam. When a PPE transits to a different PPE and the former one has insufficient dimension, the transition is penalized. Similarly, a PPE that transits to itself too often and exceeds the maximum dimension is penalized. Like location knowledge, the penalties become more severe as the distance from the dimension limit increases.

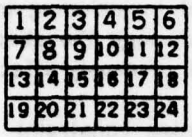
Shape measures occasionally miss an important feature of a segment because they describe shape in such abstract terms. There is a very simple solution to this problem which requires that all regions with complex shape be broken down into multiple regions that are less complex and are adjacency constrained to form the overall shape. Returning again to the cross example, the region can be broken down into five PPEs, each a 2x2 square which is easily describable with simple shape measures. These five PPEs can then be constrained so that they adjoin in a cross pattern with one in the center and four surrounding it. Of course, this technique requires extra network space and search time, so it should only be used for pathological shapes. However, it does allow arbitrarily shaped objects to be described.

3.5: SIZE KNOWLEDGE

Size is usually separate from shape. However, the cross example in the previous

section shows that the shape measure used by ARGOS also specifies information about the size of the cross. This section discusses how independent size knowledge can still be used, even with dimensional shape knowledge.

Recall that dimensional shape describes the cross as a segment with horizontal and vertical dimensions of 2-to-6 and diagonal dimensions of 1-to-4. The following region fits that shape:



Size knowledge can prevent the above region from matching this description: it has twenty-four pixels and the cross has only twenty[2]. Size is defined as an upper limit on the number of pixels in a region. PPEs that transit to themselves too often will violate the size limit and penalize their path likelihood. Like location and shape, size is extracted from the hypothesized images and used as a factor in network mergings.

Unfortunately, size is not effective for ARGOS when labeling city scenes. It will be shown in section 5.2.3 that size knowledge of this type works against correct labeling because of a quirk interaction between the search order and the nature of object shapes in natural scenes. However, this type of knowledge is still useful for Locus search in general.

There are other ways to implement size knowledge that have not yet been explored. Size can be used in the backtrace to resolve conflicts in much the same manner as shape. In addition, it is possible to implement relative size knowledge that is able to make statements like "a is twice as large as b". The implementation of this would require that the knowledge network have access to registers during the search. Various network nodes would store knowledge in these registers for use by other nodes. This scheme is similar to Augmented Transition Networks (Woods, 1970).

3.6: KNOWLEDGE CONSTRAINT

The knowledge sources that are used in ARGOS are not used independently. They are all descriptive aspects of regions, so they should work together. ARGOS currently has location, shape, and size implemented. It is reasonable to expect that the search is

2 Notice that there is a fine line separating the size knowledge from the dimensional shape knowledge; in fact they really work together.

more effective if these three are constrained to a hierarchical scheme where lower knowledge sources in the hierarchy are only accumulated if higher knowledge sources are not being violated. The highest knowledge source is location because it constrains region identification most; the lowest knowledge source is size.

Implementation of this hierarchy implies that shape information be accumulated only for nodes in the search tree that fall within the minimum bounding rectangle of the PPE. At the other end of the hierarchy, size information is only accumulated for nodes in the search tree that fall within the shape bounds. The need for this hierarchy is apparent because there is no point in including incorrectly positioned pixels in the computation of shape: it can only damage the shape measure. Similarly, inclusion of pixels that violate shape bounds will adversely affect the use of size knowledge.

In experiments, the use of knowledge constraint was not found to give significant improvements. However, ARGOS continues to use it for the slim advantage that it does yield.

3.7: CONCLUSION

Some forms of knowledge can be used with Locus more effectively than others. Since ARGOS is designed to be efficient, it does not attempt to incorporate knowledge schemes that are unnatural for Locus search. However, it is still felt that any knowledge can be used in some form.

This chapter discussed region knowledge that can be put into a network. The next chapter discusses how to make a series of networks that bring more complex knowledge to bear on the labeling of images.

CHAPTER 4: KNOWLEDGE HIERARCHIES

Up to this point, all discussion about ARGOS has centered around the labeling of Pittsburgh city scenes. A knowledge network has been described which can identify various views of the city and selected images have been successfully labeled. What about scenes of other cities such as New York? What about non-city scenes? There is a continuum of knowledge in the world and Pittsburgh is just one piece of it. The assumption that has been made is that each micro-world requires a separately designed knowledge network. This chapter is concerned with the techniques that can be used to label arbitrary scenes without the need for manual selection of networks. Although few of these techniques have been implemented, they provide some direction for future research.

The first thing that should be pointed out is that there is a reasonable upper limit to the amount of knowledge that a network can hold. Although current computer technology imposes its own limits, there are theoretical upper limits that should also be observed. Therefore, it is not reasonable to build one large network with all available knowledge. In particular, the continuum of knowledge can be divided into hierarchical levels and a network should always bridge two of these levels. This bridging effectively applies knowledge that has been acquired at a higher hierarchical level to the acquisition of knowledge at a lower level.

For the purposes of this discussion, all image knowledge will be divided into three hierarchical levels. The top level is the scene level and contains all of the scene types that can be distinguished by the image understanding system. Another way to think of this level is as the schema or frame level. Examples of scene types are: city scene, office scene, satellite scene, etc. Below this level is the viewpoint level which is concerned with the angle and distance of view. At the bottom is the object level which, given the environment and viewing position, concerns itself with correct identification of the objects in the scene.

These levels of knowledge are not the typical levels that other image understanding systems (such as Sakai et. al., 1976; Ballard et. al., 1977; Williams et. al., 1977) recognize: pixel within object within region within scene, etc. Locus networks combine those "minor" levels into a uniform structure. ARGOS hierarchies are organized along

general-to-specific lines (scene to view to object). The hierarchy spans the full depth of knowledge and is organized to extract a varying level of understanding from the entire image.

The three levels of hierarchy that ARGOS recognizes span quite a lot of knowledge, especially the top and bottom levels which are infinitely extendible in their respective directions. For example, the object level can attempt to identify the buildings in a city scene or, at a lower level of detail, all of the windows in all of the buildings[1] or, at a higher level, just the major features of the scene such as sky and mountains. Similarly, the scene level can identify a scene as "city of Pittsburgh," "city," or just "outdoor scene." There are even many distinctions to the middle category (viewpoint) such as view angle and view distance.

The rest of this chapter discusses how ARGOS could use hierarchies of knowledge to completely identify scenes. At the end is a discussion of where ARGOS currently stands in its hierarchical use of knowledge.

4.1: TODAY PITTSBURGH, TOMORROW THE WORLD

As the last chapter discussed, the ultimate source of knowledge is the internal model. This model is an actual three-dimensional description of the scene to be viewed. Since the assumption behind the use of knowledge hierarchies is that there is too much knowledge to be completely specified by one network, it will be assumed that the model is very large, perhaps a representation of the entire United States.

Before discussing the use of very large models, it is appropriate to mention their creation. ARGOS does not address the issue of automatic model generation or learning, but it would be possible to attach a feedback loop onto ARGOS which would use labeled knowledge to modify the existing model. More sophisticated learning systems could build additions to the model solely from examples found in the existing structure, but that is completely outside of the scope of this thesis. Therefore it must be assumed that the scene models are built by hand.

Hierarchical use of model knowledge always starts at the most general end of the hierarchy with identification of the scene type. Even humans start with this point of the knowledge hierarchy by first determining the "gist" of the scene (Akin and Reddy, 1976). This scene type identification process involves running the scene through a pass of Locus search with a knowledge network that contains only the major components of each scene type in the model. For example, if the model contains twenty cities, then the knowledge network used to identify the correct city might contain PPEs only for those major regions of each city that distinguish it from another.

---

1 down to the level of resolution of the sensed image.

Pittsburgh would have river and mountain PPEs and perhaps one for the U.S. Steel Bldg which is a landmark. Other cities might employ ocean, countryside, or even smog PPEs. Even the shape of the skyline can be used to distinguish cities.

Once the search is run on this network, it is easy for the machine to examine the labeled output and determine the selected city. This is because each PPE is tagged with the name of the city or view that generated it[2]. With city knowledge in hand, the machine can generate a more detailed network of the identified city which explores the scene at a lower level in the knowledge hierarchy. For example, 50 different views of the identified city can be hypothesized to form the next level network which would extract view angle and/or view distance. Note that there is no restriction on the number of Locus search iterations that can be done at a given level of hierarchy. If the world model contains 2000 cities, it might be reasonable to break the scene-type hierarchical level into two passes of Locus search; one using a network to classify them into major types (cities with tall buildings, cities near water, etc.) and then another pass with a more detailed sub-network to select from the identified types of cities.

Locus networks, therefore, form a tree structure that spans the knowledge hierarchy. In less complex world models it might be possible to pre-compute all of the networks and store them for possible use in the hierarchy traversal. In more complex models, these networks must be generated by the machine at each decision point in the identification process.

Extremely complex models have the interesting property that the lower (more specific) levels of knowledge networks all look the same. For example, once the proper city and building has been identified, the details of the building are very similar to the details of any other building. These models no longer have a tree structure to their knowledge hierarchy: it is now a network structure that re-combines knowledge paths at the lower levels. Add to this observation the fact that there is bound to be noise and error in the hierarchy searching process, and a fascinating observation is made: *Locus search can be used to guide the selection of knowledge networks.* All of the efficient, non-backtracking search can be used at a much higher level: the Primitive Knowledge Network level. So Locus search can be used both for labeling pixels within images and for labeling images within world models.

Is there a higher dimension that Locus can run on? It is possible to envision hierarchies of world models which vary according to the *intent* of the model creator. A city planner would build a world model that reflects the populated vs. unpopulated areas of cities because the planner is concerned with growth trends. A military tactician would build a model that contains only strategic points, be they telephone switching centers or harbor docks. So there is a higher level of knowledge that sits on top of the hierarchies discussed in this chapter. This level deals with the actual semantics of an image and can indisputably be called "image understanding."

---

2 Even those PPEs that are merged in the network reduction process can retain ambiguous information about their source.

4.2: KNOWLEDGE USED BY ARGOS

Needless to say, ARGOS is not able to analyze any scene in terms of arbitrary levels of understanding. In fact, ARGOS is barely able to transmit results from one level of the knowledge hierarchy to another. It does demonstrate the use of hierarchical knowledge with a simple task. The task that was selected was highly dependent on the images that were available. Since the fifteen images were all of Pittsburgh from five different views, the obvious hierarchical tasks were view angle identification followed by object identification.

View angle identification consists of building a network with many views of the city. This task used 24 views at 15° intervals around the city. Prior to network construction, the number of significant objects on the city was reduced from 58 to 16 (see Appendix II). This is because the view angle task is more general than the knowledge base which was selected for the object identification task. The selection of 16 PPEs was done automatically by examining the 24 machine-generated views of the city and counting the number of times each building contributed to the skyline. Any building which formed part of the skyline in over 50% of the views was kept as a separate PPE. The other buildings were eliminated from the list and merged into the Miscellaneous Buildings PPE.

To determine the angle of view for each photograph of the city, ARGOS labeled each image and extracted the most popular angle from the labels (recall that each label includes information about the original view or views from which it came). ARGOS typically selected one or two angles and the values were in error by an average of 30° for the training images and 50° for the test images.

The next level of the knowledge hierarchy is the object identification level. Presumably, ARGOS should be able to use the predicted angles from the view angle runs and use that knowledge to improve the object identification. A number of schemes were tried, none of which yielded significant improvement in the object recognition.

The first experiment in knowledge hierarchy traversal involved building new networks using the predicted views from the view angle task. This sounds like a good thing to do, but it constrains the search too tightly. Also, if there is any error in the predicted view angle, then the search is totally destroyed because the global path is guaranteed to be missing from the knowledge network.

The next step involved using the error of the view angle task as a guide in building the object identification networks. Instead of networks built only from the predicted view, ARGOS built networks from the predicted view plus a range of 45° on either side. Thus the average view recognition error was used to determine a range of views in the object identification task. This still refused to yield worthwhile results because the network used in object identification needs to be rich in transitions.

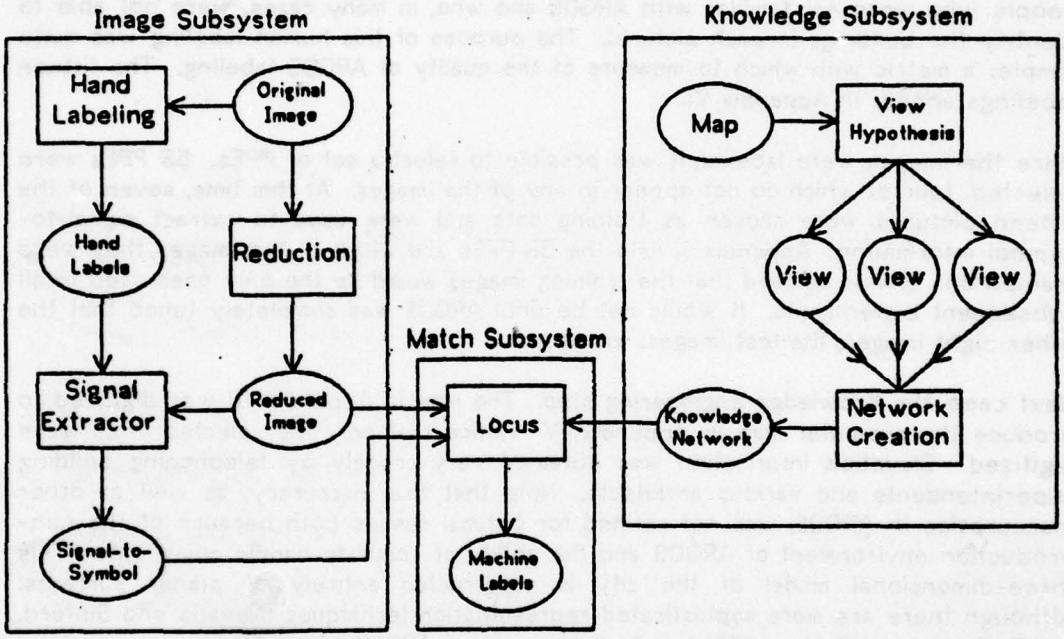
A completely different approach was then tried. The selected view angles were implemented as a knowledge source that penalizes transitions to PPEs that are not in the proper set of angles. For example, if a photograph was estimated by ARGOS to

have been taken from 120°, then the object identification task would penalize transitions to PPEs that are not tagged with angles between 75° and 165° (remember the 45° error). This technique not only yields the best results, but it has the added advantage that the object identification network does not have to be re-built for each image. The only disadvantage is that the object identification network is fairly large since it must contain knowledge about the entire view angle identification task.

The surface has only been scratched with respect to hierarchies of knowledge. Future results can be expected to be much better. The next chapter discusses the precise environment of the testing that was done with ARGOS. In spite of many noisy and inaccurate sources of knowledge, ARGOS performed well in the labeling of real-world scenes.

CHAPTER 5: RESULTS

This chapter describes a series of experiments that were made with the ARGOS image understanding system. The system as depicted below has three main parts: the image subsystem, the knowledge subsystem, and the match subsystem.



The image subsystem of ARGOS was described in Chapter 2. It consists of a series of programs that reduce large images to a workable size and accumulate useful feature vectors for each pixel. In addition, this subsystem allows interactive segmentation and labeling of the sensed scenes for use in result evaluation and for the extraction of signal-to-symbol information.

The knowledge subsystem creates networks. It contains programs for building the city model from maps, sketching views of the city model, extracting knowledge from the views, and building networks from the extracted knowledge.

The match subsystem is the program that uses Locus search to label images. It takes signal and signal-to-symbol data from the image subsystem and matches that with networks from the knowledge subsystem to generate labelings of images.

The remainder of this chapter describes the test environment in detail and examines some of the more interesting results that were obtained.

5.1: INPUT

In the beginning, there were fifteen pictures taken from five different vantage points about the city. These images were segmented and labeled by untrained humans (i.e. people who were not familiar with ARGOS and who, in many cases, were not able to identify the buildings in each picture). The purpose of this human labeling was quite simple: a metric with which to measure of the quality of ARGOS labeling. The fifteen labelings appear in Appendix VI.

Once the images were labeled, it was possible to select a set of PPEs. 58 PPEs were selected, four of which do not appear in any of the images. At this time, seven of the fifteen pictures were chosen as training data and were used to extract signal-to-symbol information. Appendix II lists the 58 PPEs and which of the images they were trained on. It was decided that the training images would be the only ones used in all subsequent experiments. It would not be until ARGOS was completely tuned that the other eight images, the test images, would run.

Next came the knowledge engineering step. The map in Appendix III was digitized to produce the computer map in Appendix IV. Notice that only the selected PPEs were digitized. Elevation information was obtained very crudely by telephoning building superintendents and various architects. Note that this inaccuracy, as well as other inaccuracies in ARGOS, was not refined for optimal results both because of the non-production environment of ARGOS and the ability of Locus to handle noisy data. This three-dimensional model of the city is constructed entirely of planar surfaces. Although there are more sophisticated representation techniques (Nevatia and Binford, 1977; Reddy and Rubin, 1978) this technique serves ARGOS adequately.

With a machine model of the city, "hypothesized" views could be made (see Appendix V for examples). The amount and type of views varied with the task. Region adjacency, location, shape, and size knowledge was extracted from the hypothesized views and a knowledge network was built. ARGOS was now ready to run.

5.2: EXPERIMENTS

ARGOS has many parameters which must be tuned to get optimal performance. Some, such as the pruning cutoff, have already been mentioned in previous discussions and others are of little interest. This section describes three of the more interesting experiments which answered these questions:

1) What knowledge should be considered in the merging of PPEs from different hypothesized views during network creation?
2) How much of the PPE merging should be done during network creation?
3) Why is size knowledge not useful?

5.2.1: Use of Location and Shape in Network Reduction

Three algorithms were proposed for the merging of multiple instances of PPEs from different hypothesized views (see section 3.1). Each algorithm allows PPEs to combine if a certain percentage of their region adjacencies are identical. The algorithms differ only in the amount of consideration they give to location and dimensional shape knowledge[1].

The first algorithm uses no location or shape knowledge. It merely compares the adjacency knowledge and merges the PPEs if the percentage of their identical adjacencies is above the threshold for merging. Recall the second example of section 3.1. The Alcoa Bldg appeared in front of the U.S. Steel Bldg alternatively to the right and the left in two hypothesized views. Using only horizontal and vertical adjacencies, the Alcoa PPEs in these two views share 50% of their adjacencies (each agrees on the vertical and disagrees on the horizontal). If a network is built that allows an adjacency reduction threshold of 50% or less, then the two Alcoa PPEs will be merged (as they were in the example in section 3.1). Notice that this merging algorithm has not considered the shape or location of the PPE regions: it merges solely on the basis of adjacency similarities.

The second algorithm depends highly on location and shape knowledge. Not only must the two PPEs being considered for merging have adjacencies that are identical to a fixed percentage (as in the first algorithm), but the centers of the minimum-bounding-rectangles (as defined by location knowledge) must be within the same percentage of the image size. In the above example, if the centers of the minimum-bounding-rectangles of two Alcoa PPEs were more than 50 pixels apart in the horizontal direction, then 50% reduction would allow merging on the basis of adjacencies but would prevent the merge on the basis of location. This is because the image is 100 pixels wide and the regions are more than 50% of that apart. In addition to using location knowledge, this second algorithm requires that the minimum and maximum

1 It will be shown in section 5.2.3 that size knowledge is not useful. Therefore, it is not considered here.

dimensional shape limits (in all four directions) be within tolerance to each other. By tolerance is meant that the smaller value must be at least the fixed percentage of the larger value. Returning to the example, if the two hypothesized views are from such different distances that the dimensional shape bound of one is more than twice the size of the other, then the merging will not occur.

The third algorithm uses the same technique as the second algorithm except that it relaxes the restrictions on shape. Only maximum dimensional shape is compared, and only in the horizontal and vertical directions (not in the diagonal directions). The effective result is that the sizes of the minimum-bounding-rectangles are compared. This last reduction algorithm was consistently found to be superior to the others in all of the ARGOS tasks.

5.2.2: Determination of Network Reduction Amount

Using the selected reduction algorithm from the previous subsection, eleven networks were built with varying percentages of reduction. These networks were generated for the object identification task and contained enough views to label all of the city images. The following table shows the percentage of reduction, the number of PPEs in the reduced networks, and the branching factor of each network. The branching factor is the average number of arcs (in each direction) emanating from a typical PPE.

Reduction	PPEs	Branching Factor
100	274	3.1
90	273	3.1
80	259	3.1
70	231	3.2
60	183	3.5
50	132	3.8
40	114	4.0
30	95	4.4
20	77	4.6
10	66	5.0
0	58	5.6

The first training image was run on all of these networks. The larger networks were unable to label this training image because the beam (which allowed 75 paths for this experiment) could not hold all of the independent paths. Only the lower three reductions gave good results, so all of the training images were run on them.

The 0% and 10% networks gave relatively equal results, both of which were slightly better than the network with 20% reduction. The choice, then, is between 10% reduction and 0% reduction. However 0% reduction, by definition, does not allow extraction of the identified hypothesized view since all hypotheses of the same region are merged. Therefore the 10% reduction is considered best for this object identification task.

5.2.3: Rejection of Size Knowledge

In all experiments that used size knowledge, the results were worse than without it. The following explanation is offered.

When the location and dimensional shape of a region have already been specified, size knowledge serves only to prevent the region from "filling out" its prescribed area. Since Locus works in raster scan from top to bottom, only the excess pixels near the bottom of the region will be penalized because they are the first pixels that exceed the maximum size. Therefore, size knowledge is useful in defining regions that are large near the top and get smaller near the bottom because it penalizes the bottom area pixels and forces recognition to "slim down". A quick look at the pictures in Appendix I, however, will convince anyone that most of the regions in this domain are shaped exactly the opposite: large on the bottom and small on the top. Therefore, size knowledge only serves to reject regions that would otherwise be correctly labeled.

Size knowledge is wrong for this task and this search order. However, it is not necessarily wrong for all uses of Locus search.

5.3: UNSEGMENTED SYSTEM: RESULTS

In addition to the experiments described in the last section, there were the following experimental results:

> The per-pixel penalty for violations of location knowledge was determined to be 5 (on a scale of 0 to 255, see section 2.4.3.7). This means that PPEs that are matched to pixels which are n pixels outside of the location boundary will have their path likelihood penalized by 5n. Penalties less than this did not adequately enforce location knowledge and penalties greater than 5 damaged the search by decreasing the importance of other factors in the path likelihood equation.
>
> The per-pixel penalty for violations of dimensional shape knowledge was determined to be 5 for similar reasons of relative weight in the path likelihood equation.
>
> The relative weight of optical matches was determined to be 200. This means that the difference between the best and worst match of a PPE to a pixel is 200. Thus, it takes other factors totalling over 200 in value to override a signal-to-symbol match in the path likelihood equation.
>
> The optimal size of the beam was found to be 25 entries. This is the smallest beam size that the system could use without damaging the search.

It was determined by reducing the size until the labeling quality started to decline.

The criterion for beam pruning was set to 100 below the current best beam entry. Thus, a candidate for beam entry is rejected if its value is more than 100 greater than the current best member of the beam. Cutoffs greater than 100 allowed too much into the beam and confused the search. Cutoffs less than 100 pruned too severely and totally damaged the beam.

Once tuned, ARGOS labeled the seven training images and eight test images shown in Appendix I. These runs used unsegmented images and performed an object identification task. In an attempt to provide a more global evaluation of these results, the system compared segments in the output of ARGOS to segments in the human labeling. It then computed the percentage of the image that was labeled correctly. For example, if a machine labeled pixel does not agree with the human label but is part of a larger segment of machine labels that do agree, then that pixel is considered to be correct. This evaluation avoids minor inconsistencies in the human labeling.

Prior to the above evaluation, all of the output of ARGOS is smoothed with three separate operations. The first step is simple smoothing (Ejiri, 1971) which takes any single pixel that is surrounded by many pixels of a different label and changes the label on the center pixel to that of its environment. The second step throws out any region with less than eight pixels. These pixels become unlabeled because they do not form a large enough area to have meaning. The third step fills in any unlabeled holes in a region. The unlabeled area must be completely surrounded by pixels of the same PPE to be filled in with that label. Since there are no "containership" relationships in the Pittsburgh images, this operation serves only to fill gaps in the labeling.

The numbers below do not consider unlabeled pixels. Instead, they are the percentage of the labeled area that is correct. See Appendix VI for these machine labelings and the human labelings of all fifteen city scenes.

Training Image	% Correct	Test Image	% Correct
1	77	1	67
2	89	2	84
3	83	3	78
4	73	4	65
5	69	5	87
6	46	6	90
7	94	7	84
		8	15
Overall Average	76		71

The overall average labeling quality is 73%. If the two close-up scenes (training scene 6 and test scene 8, both of which scored badly) are removed from these statistics, then the training scenes achieve 81% accuracy and the test scenes achieve 79% accuracy. Overall accuracy on thirteen of the Pittsburgh city scenes is 80%.

5.4: PRE-SEGMENTED SYSTEM: RESULTS

The version of ARGOS which uses pre-segmented images ran with many of the same parameter settings as the unsegmented system. The task, however, was different. Instead of training on the existing images to obtain high labeling accuracy, the pre-segmented system implemented the two-level knowledge hierarchy discussed in Chapter 4. For each image, there is a view angle identification run and an object identification run. The following table shows the results of the hand-segmented images in the view angle identification task:

Image	True Angle	ARGOS Angle	Error
Training 1	300-315	300	0
Training 2	300	330-345	30
Training 3	240-255	330-345	75
Training 4	0-15	15	0
Training 5	0-15	330	30
Training 6	345	0	15
Training 7	45-60	330-345	60
Training Average			30
Test 1	315	195	120
Test 2	285-300	330-345	30
Test 3	300-315	240	60
Test 4	255	240	15
Test 5	45	0	45
Test 6	60-75	135	60
Test 7	60-75	0	60
Test 8	15-30	0	15
Test Average			51
Overall Average			41

Using a "favorite view" penalty of 25 on all transitions more than 45° from the predicted view shown above, the following results were obtained in the object identification task. The actual output of test scene 5 is shown in Appendix VII. Note that the numbers below measure the percentage of the image that is labeled correctly, not the percentage of the labels that are correct. Thus, the numbers are somewhat lower than those presented in the previous section.

Image	% Correct	% Incorrect
Training 1	72	12
Training 2	74	7
Training 3	70	10
Training 4	49	22
Training 5	58	19
Training 6	37	54
Training 7	59	26
Training Average	60	21
Test 1	27	44
Test 2	35	53
Test 3	41	42
Test 4	47	26
Test 5	79	11
Test 6	37	33
Test 7	19	64
Test 8	20	73
Test Average	38	43
Overall Average	48	33

In addition to hand segmentations, ARGOS interpreted machine segmentations of the same images. These segmentations were obtained by a clustering algorithm (Shafer and Kanade, 1978). The following table shows the results of the view angle identification task.

Image	True Angle	ARGOS Angle	Error
Training 1	300-315	345	30
Training 2	300	330-345	30
Training 3	240-255	0	105
Training 4	0-15	120-135	105
Training 5	0-15	330	30
Training 6	345	330	15
Training 7	45-60	345	60
Training Average			54
Test 1	315	30	75
Test 2	285-300	330-345	30
Test 3	300-315	0	45
Test 4	255	255	0
Test 5	45	330	75
Test 6	60-75	345	75
Test 7	60-75	300	120
Test 8	15-30	135	105
Test Average			66
Overall Average			60

Using a favorite view penalty on all transitions which are more than 60° from the predicted view shown above, the following results were obtained in the object identification task. The actual output of test scene 5 is shown in Appendix VIII.

Image	% Correct	% Incorrect
Training 1	40	33
Training 2	71	17
Training 3	40	41
Training 4	2	44
Training 5	42	22
Training 6	35	48
Training 7	55	15
Training Average	41	31
Test 1	27	56
Test 2	31	49
Test 3	43	33
Test 4	43	43
Test 5	41	38
Test 6	34	44
Test 7	24	51
Test 8	19	74
Test Average	33	49
Overall Average	37	41

The machine segmentations were produced in the last weeks of this thesis research, so there was no time to tune the system for them. These poor results are presented for completeness sake only and should not be regarded with any disappointment.

CHAPTER 6: CONCLUSION

In the previous chapters, Locus search was presented as a technique for image understanding. Discussions demonstrated how knowledge can be added to the search and how hierarchies of knowledge can be organized around the search. Finally, experiments were described in which ARGOS labeled photographs of the city of Pittsburgh.

This last chapter cleans up the thesis by discussing the results obtained from ARGOS. It is as important to know why it fails as it is to know why it works. The chapter finishes by summarizing, discussing, and concluding the thesis.

6.1: ERROR ANALYSIS

Three types of errors were made by ARGOS: errors of scale, errors of position, and errors of shape. The errors of scale were caused by the varying sizes of objects in the images. For example, the Hilton Hotel ("HI" in the appendices) ranges in size from 9 pixels across in training scene 5 (page A19) to 58 pixels across in test scene 8 (page A29). The errors of position include location knowledge errors and adjacency errors. The later were the most damaging: all it took was one region out of place, and a half dozen other regions were incorrectly labeled relative to the original erroneous one. The errors of shape can be seen in all of the labelings as diagonal zig-zags on the region borders.

Before discussing these classes of errors, it is appropriate to mention the successes that ARGOS achieved. The system did best on its identification of major parts of the image. The sky, the mountains, and the three rivers were almost always identified correctly. This is due to the ease of identification on the signal and the symbol level.

Many of the apparent labeling mistakes were entirely reasonable. For example, the identification of Miscellaneous Buildings (B) in the middle of the river which occurred in training scene 2 and test scenes 1, 2, 3, and 5 (pages A16, A22-A24, A26) is entirely reasonable since those areas of the river actually contain reflections of buildings.

6.1.1: Scale Errors

Very often, labeled regions were in the correct place but did not cover the entire original region. For example, notice the labeling of One Oliver Plaza ("OO") in test scene 3 on page A24. The human labeling of the image indicates that this region is about 10 pixels wide and 20 pixels high. The machine labeling, however, shows the building labeled with two small regions which together cover only 45 pixels. All of the other pixels in that area are unlabeled which means that, although the region was correctly identified, only 25% of it was labeled.

One reason that this area was labeled with such small regions can be found in the shape knowledge. The dimensional shape bounds for One Oliver Plaza are 1-to-10 horizontally and 2-to-40 vertically. Although this allows a 10 by 40 pixel region to be labeled, it also allows much smaller regions to go unpenalized.

Two cures can be found for bad shape knowledge. The first cure requires that hypothesized views be smoothed before knowledge is extracted. This way, jagged and pointed edges of regions would not affect the dimensional shape counts. The second cure involves more severe pruning of the data accumulated from the hypothesized images. If, for example, only one of the lines through an object has a pixel count below 5 and all other lines pass through at least 10 pixels, then the lower bound of the dimensional shape should be 10 and not 5. This sort of extreme case rejection is done to a modest extent now, but it should probably be done more.

Another reason that regions are labeled with incorrect size is that the hypothesized views try to mimic the images but often fail. For example, notice that the Hilton Hotel ("HI") in test scene 8 (page A29) is nearly 60 pixels wide. The dimensional shape knowledge for Hilton Hotel penalizes regions wider than 36 pixels because it is unfamiliar with such a close-up view. So the problem of bad shape knowledge is also caused by bad hypotheses. This problem would not be as severe if ARGOS used a full knowledge hierarchy because the size variations would be expected by the viewpoint knowledge networks and would be correctly hypothesized after the viewpoint level had run.

Another type of size error that was encountered is over-labeling. This happened in the distant views from the West (training scenes 4 and 5). For example, notice the labeling of the U.S. Steel Bldg ("US") in training scene 5 (page A19). The labeled area not only covers the correct region, but spills over into about a half dozen other neighboring regions. It appears that the same problem of incorrect hypothesis has caused this size error.

6.1.2: Position Errors

How did a region of Miscellaneous Buildings ("B") in training scene 1 (page A15) get the label Monongahela River ("MR")? One reason, which must hold true in all

mislabelings, is that they do have similar characteristics. The other reason for the mislabeling has to do with a sequence of bad adjacencies. Observe the labeling on the left side of training scene 1. Once the Sixth Ave. Parking Garage ("SP") was labeled Miscellaneous Buildings, all of the regions above that got labeled incorrectly. The first error above that is the labeling of the Pick Roosevelt Hotel ("PI") as Snow ("SN"). Both are bright regions, so the mistake is not unreasonable. Finally, the top region in this area is labeled Monongahela River. Again, it is not an unreasonable guess, but all three are wrong. It should be noted that the Monongahela River is legal above Snow which is legal above buildings. Thus, this three-region error is locally consistent which helps to reinforce the error.

Adjacency errors are started by bad signal-to-symbol matches and reinforced by local adjacency constraints. Although Locus is supposed to enforce global constraints, it is tuned to allow noise in the search which, in turn, generates local errors. The only real problem with these errors is that they propagate and cause large areas to be incorrectly labeled.

Bad location knowledge is another source of error. The problem is that location knowledge is obtained from the hypothesized views and these views seldom align correctly with the city images. As a result, location knowledge which is not completely generalized is often too specific AND WRONG. The results of bad location knowledge can be seen in the pre-segmented images (Appendices VII and VIII) where a few buildings (most notably Gulf ("GU")) were labeled far too frequently. This is because most of the other PPEs had incorrect location knowledge causing their transition likelihoods to be heavily penalized.

6.1.3: Shape Errors

The most distracting aspect of the unsegmented machine labelings is the irregularity of the labeled region shapes. These regions have as many diagonal edges as they have horizontal and vertical edges, yet the regions in the city scenes and the hypothesized views have very few diagonal edges.

One obvious reason for this is that all four directions of adjacency are given equal weight during the search. Since all horizontal and vertical adjacencies can also adjoin diagonally, it should be expected that an over-abundance of diagonal edges will be found. This can be solved by determining an overall preference for each direction and using it as a knowledge source. However, the real reason for most of the shape errors is only marginally connected with the use of diagonal adjacencies. The culprit is backtrace conflicts.

During the backtrace, conflicts occur when the four surrounding pixels disagree over a label assignment. Conflicts are not always resolvable, so many pixels are left unlabeled. However, an unlabeled pixel does not contribute to the labeling of its neighbors: the selection passes through the unlabeled pixels and in many cases returns

to the conflict producing neighbors. Thus, it is an artifact of Locus that one conflict point can "generate" lines of unlabeled pixels emanating from the conflict in the four directions of adjacency. These lines are the major cause of zig-zag regions.

One way to eliminate this problem is to disregard backtrace neighbors that are too far away. Another solution requires more frequent conflict resolution. A third possibility is simple smoothing of irregular edges. However, all of these solutions ignore knowledge constraints, so it can be expected that they will reduce labeling quality. The only real solution is to completely re-evaluate the backtrace, perhaps adding some knowledge and search to it.

In the absence of a good solution to the problem of shape irregularities, it is possible to smooth the regions. This would yield regular regions and would probably enhance labeling quality.

6.1.4: Future Exploration

This section has pointed out a number of improvements that should (and probably will) be implemented. Shape and location knowledge must be obtained more intelligently. View hypothesis should be done with more of an eye towards the knowledge that is to be used (i.e. generated views should make meaningful knowledge easy to extract). A better low-level system is needed to be able to distinguish the many different PPEs. And, of course, PPE adjacency knowledge should be used more effectively, both on the forward pass and the backtrace.

6.2: SUMMARY

This thesis has described the ARGOS image understanding system. ARGOS is able to use knowledge from maps of downtown Pittsburgh in the interpretation of photographs of the city. It is this use of knowledge that distinguishes image understanding from image interpretation.

Application of knowledge requires that the knowledge and the image be represented with common units so that they may be matched. This common unit, called a Primitive Picture Element or PPE, is the atomic image label. All regions of the image have unique PPE labels and all components of the knowledge have unique PPE labels. Therefore both the image and the knowledge are expressible with this common unit.

The knowledge used by ARGOS is organized into networks of PPE nodes. A connection between two nodes indicates a physical adjacency of those PPE regions in the image.

Information is associated with the connections specifying the nature of the adjacency. In addition, information is associated with each PPE node that describes the size, shape, location, and optical characteristics of the region in the image. It is the task of ARGOS to find the correct path through the network which corresponds with each image. This path is equivalent to a labeling of the image.

Determination of the correct network path is done with Locus search. Locus is a powerful search technique that was used successfully by the Harpy speech system (Lowerre, 1976). Locus is called a "beam search" because it builds a highly pruned search tree of labeling alternatives which resembles a beam. Each step along the beam corresponds to a different part of the image, and the various PPE possibilities at each step are the possible label alternatives for that part of the image. Determination of beam entries is based on a recursively defined "path likelihood" which contains information about (1) the absolute similarity between the PPE characteristics and the characteristics of that part of the image, (2) the network connections to that PPE node, and (3) previous entries in the beam.

The third component of the path likelihood equation is the recursive aspect of the beam. It gives Locus a first-order Markov flavor and it allows the beam to be constructed in one pass through the image. It is one of the significant contributions of this thesis that some of the first-order Markov nature is retained even though the image is not linear. The fact is that there is no way to scan through a two-dimensional image that will allow each new part of the image to be adjacent to the previously examined part and no other. This topology problem causes the Markov assumption to be heuristically modified so that *all adjacent* areas in the image are considered to be the immediately previous node.

Given that a beam of node connections has been built with only one scan of the image, a backwards scan (the backtrace) then examines the beam in reverse order and selects a unique labeling from the beam options. This labeling contains global constraint knowledge since it is backing up a recursive chain and therefore can select on the basis of knowledge about the entire chain. The final labeling result is only sub-optimal because of the pruning that is done in the creation of the beam. This pruning is a heuristic of Locus search that allows great space and time savings at a negligible cost in accuracy.

The backtrace of Locus search also suffers from the topological problems of images. This thesis has explored a number of options for resolving the path inconsistencies that are caused by multiple adjacencies during the backtrace. Most of these resolution techniques are able to work with local knowledge, but some proposed alternatives require global information about the image.

The thesis has also explored the need to use many networks and many "passes" of Locus search in the complete identification of an image. It is expected that a level of visual knowledge on the order of a human's would require hierarchies of knowledge networks which would successively narrow the interpretation options down until a labeling was found. The levels of analysis would proceed from scene identification to viewpoint identification to object identification. It is even expected that a form of Locus search could be used to guide the selection of these knowledge networks.

In addition to the abstract presentation of Locus, the thesis has presented the experimental results of ARGOS. When given city maps and training images, the system was able to identify major components of pictures of downtown Pittsburgh. It not only was able to select the proper viewpoint but it achieved less than 20% error by area in the labeling of totally uncontrived photographs of the city.

6.3: DISCUSSION

The graph structure representation of Locus is a natural outgrowth of work in languages (Aho and Ullman, 1972), graph representations (Harlow, 1973), and syntax directed pattern recognition (Narasimhan, 1966; Clowes, 1969; and Fu, 1976). The Locus approach principally differs from the above in how the network representation is to be used. It rejects the notion that image recognition is best viewed as a problem in parsing. Given the error and uncertainty associated with the decisions, the problem tends to be not one of deciding whether a given pattern is parsable but rather one of search, i.e., deciding which of the many acceptable alternative paths represents the near-optimal interpretation.

The view that the problem of image recognition is one of constraint satisfying search has been gaining increasing acceptance (Waltz, 1975; Tenenbaum and Barrow, 1976; Rosenfeld, Hummel and Zucker, 1976). Locus also subscribes to this viewpoint and differs mainly from the other efforts in the representation of constraints and the method of search.

The realization that it is important to introduce some measure of the degree of uncertainty into the interpretation process is reflected in the papers by Fischler and Elschlager (1973), Feldman and Yakimovsky (1975), and the probabilistic relaxation methods at SRI and the University of Maryland. Locus is able to handle search in the presence of error and uncertainty in a natural and straightforward manner provided all knowledge and constraints are represented in terms of a graph structure.

The best-first search given by the A* algorithm (Nilsson, 1971) and the breadth-first graph search of the dynamic programming algorithms (Levine, 1977; Bellman and Dreyfus, 1962) provide alternative approaches to optimal graph search problems. The beam search technique of the Locus model provides a minimal effort near-optimal solution. It appears to be effective in cases where the evaluation function is dependent on an external signal source and where a large number of decisions are related to each other as they attempt to provide alternative interpretations of the same signal segment.

ARGOS is significant because it is able to label highly complex scenes. Although much work lies ahead for image understanding by computer, experience with ARGOS adds another step to our understanding of the application of knowledge, and ultimately to our understanding of human vision and the human brain.

6.4: CONCLUSION

This thesis has made two contributions: the modification of Locus search for image understanding and the development of ARGOS as a working image understanding system. Each of these contributions encompasses a number of smaller results and observations.

Modification of Locus involves a heuristic to handle the two-dimensionality of images. Implementation for the speech task used a linear search tree that could be built and backtraced in a well-defined manner. The first-order Markov nature of Locus allowed each element of this tree to be dependent on the one previous element in a strict recursive manner. In ARGOS, this first-order Markov assumption must be changed by the complexity of the signal. Locus now becomes an "adjacency first-order" Markov system that uses *all* surrounding elements both on the forward pass and the backtrace.

Modification of the forward pass involves the inclusion of all adjacency directions in the path likelihood equation. This is done by averaging each direction's contribution. In images that are broken down into a grid of pixels, there are four neighbors to be considered (assuming a raster scan and physical adjacency of all neighbors). Images that are segmented into varying sized and shaped areas have a varying number of neighbors located at arbitrary positions in the search tree.

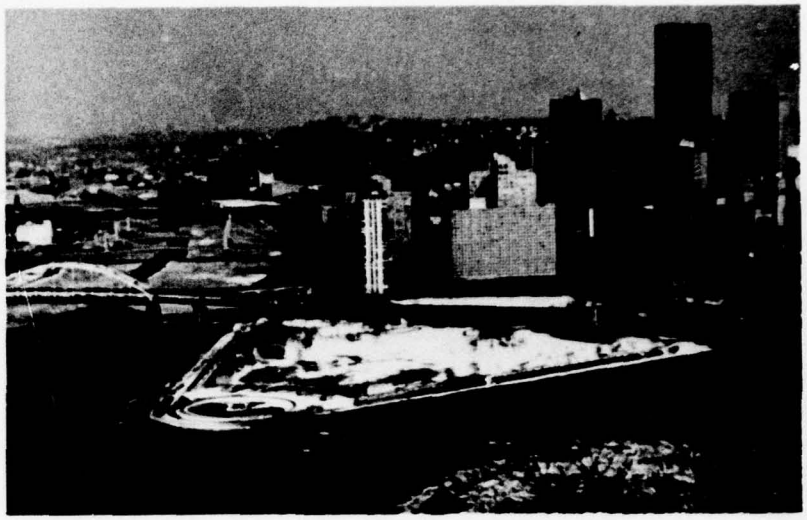
The backtrace also runs into issues of image topology. Because there are a varying number of neighbors in the forward pass, the backtrace follows a varying number of paths as it climbs the search tree. These paths frequently join together with differing results, generating conflicts in the global path. A number of local conflict resolution techniques were explored and some global methods were proposed. Those covered in the thesis are: relational network consistency, directional preference, voting, and the use of size, shape, and location knowledge.

In addition to the basic Locus search, the thesis has explored the addition of various knowledge sources. Using a three-dimensional model to generate hypothesized two-dimensional views, ARGOS is able to extract object location, size, and shape. In addition, it is possible to hypothesize the two-dimensional views in a hierarchical order starting at scene selection and proceeding to viewpoint selection and finally object identification. It is believed that this hierarchy can be used to apply massive amounts of knowledge to the interpretation of images.

The final result of this thesis is, of course, the output of ARGOS. The system was trained on seven images of downtown Pittsburgh. Another eight were saved for test purposes after the system was tuned. The training images were labeled with 58% accuracy and 18% error on a pixel basis (some pixels were left unlabeled). The test images were 48% correct and 22% incorrect. In addition, ARGOS was able to correctly identify the major components of almost every image (only three of the fifteen showed a lack of "understanding" by the system). In a view angle selection task, the system identified the angle of view with an average error of 40^{o}. ARGOS has therefore demonstrated the applicability of Locus search to the image understanding task.



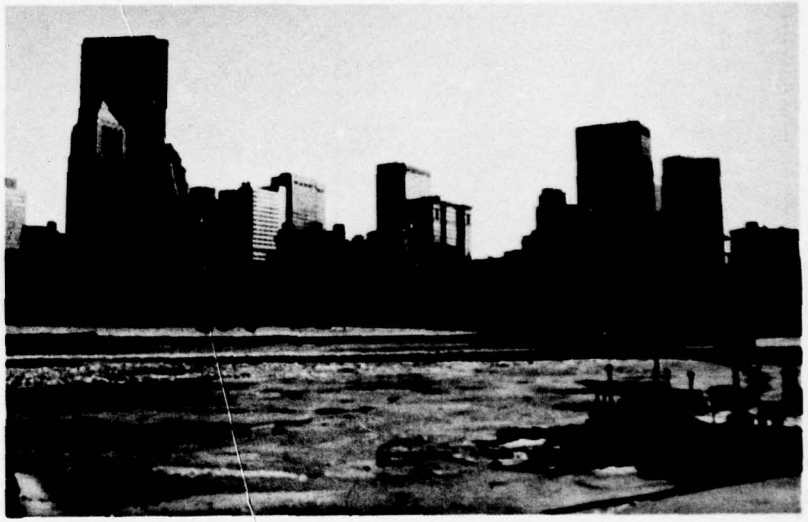
Training Scene 2



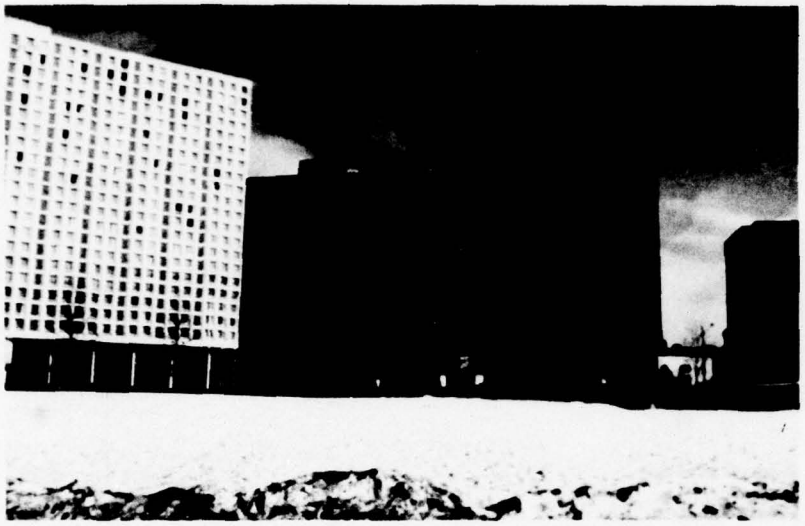
Training Scene 4



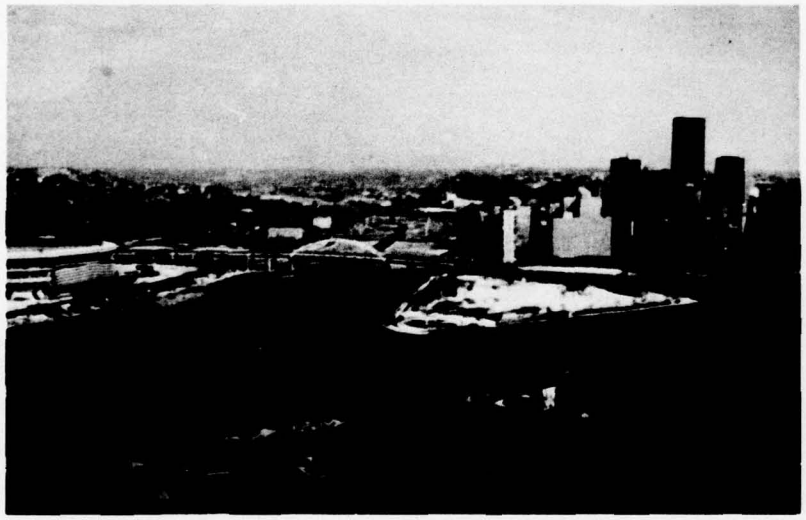
Training Scene 1



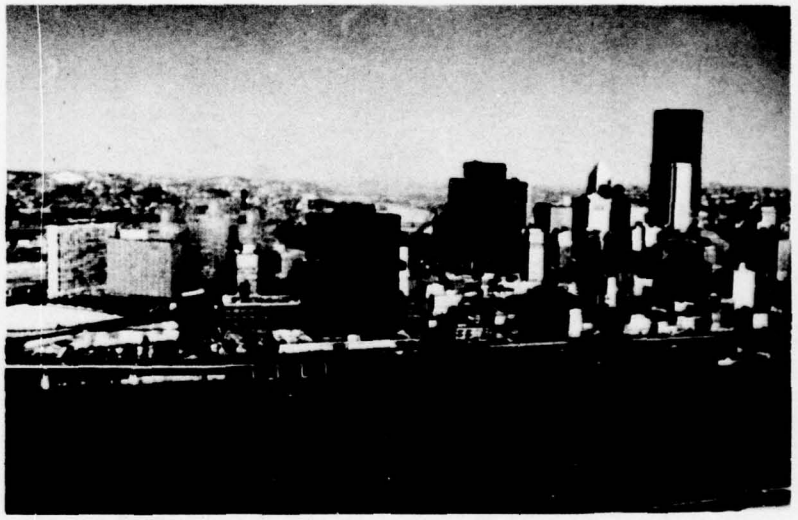
Training Scene 3



Training Scene 6



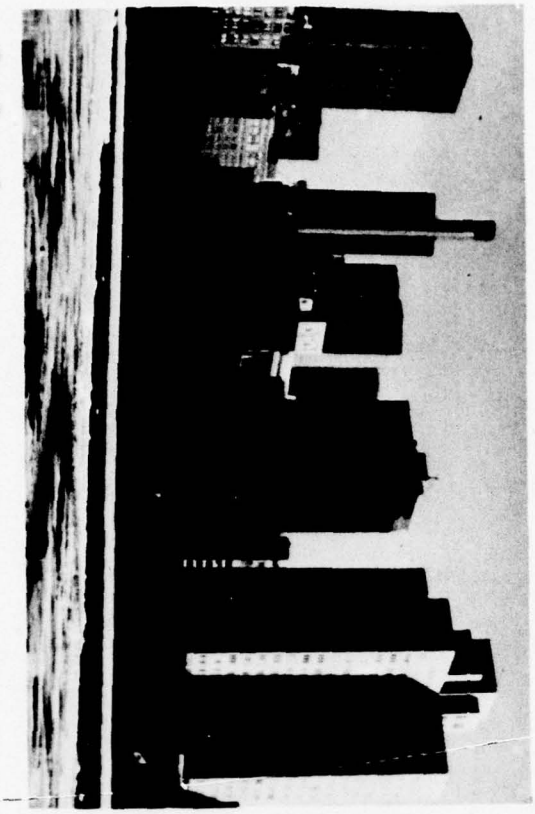
Training Scene 5



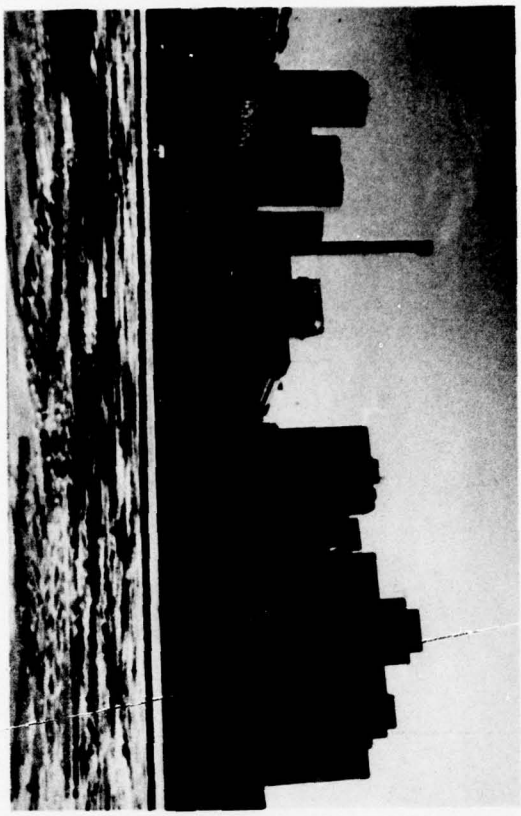
Training Scene 7

APPENDIX I: Original Photographs (continued)

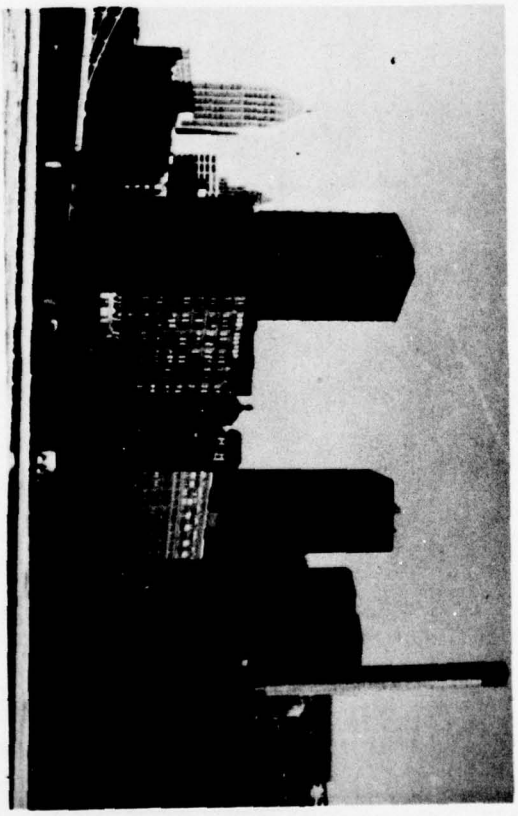
Test Scene 1



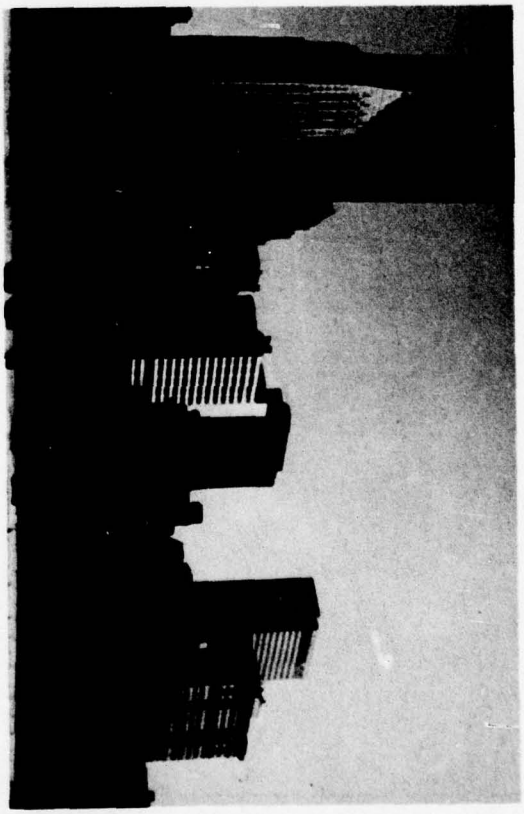
Test Scene 2



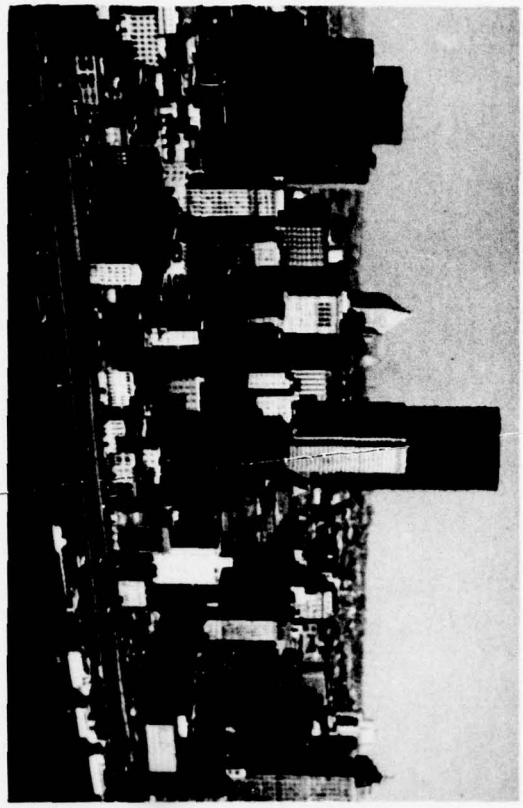
Test Scene 3



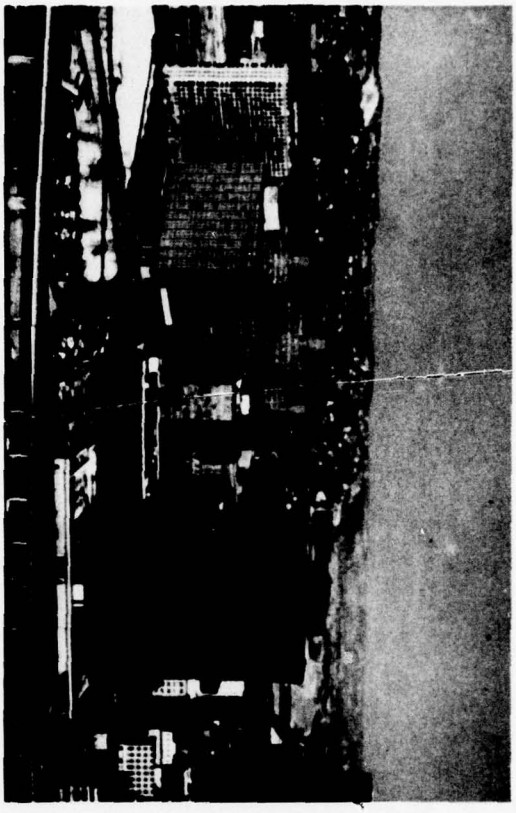
Test Scene 4



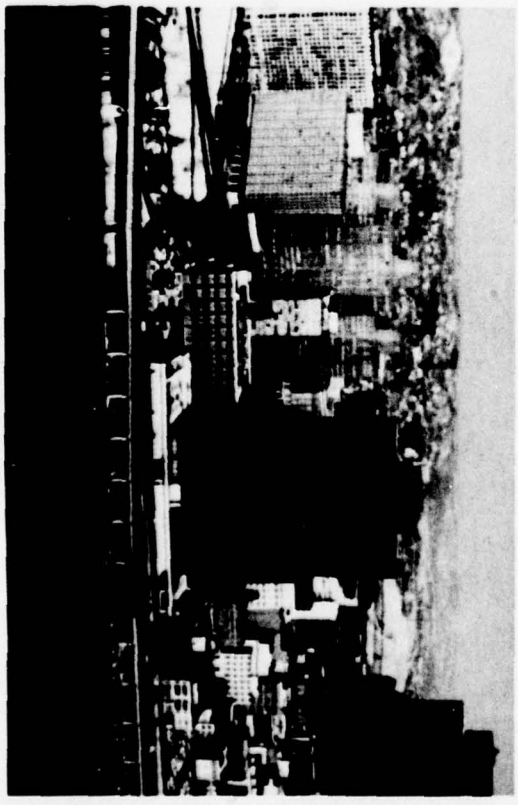
Test Scene 5



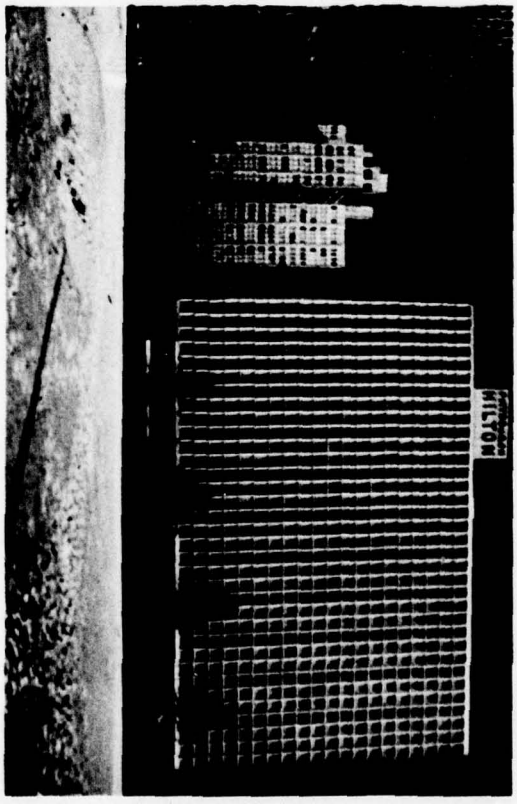
Test Scene 6



Test Scene 7



Test Scene 8



APPENDIX II: Primitive Picture Elements

This table shows all of the primitive picture elements (PPEs) that were selected for the object identification labeling of the Pittsburgh images. The symbol column is the identifier used in the labelings in Appendices VI, VII, and VIII. The training image column reports which of the seven training images were used to extract signal-to-symbol matches for that PPE. Note that four of the PPEs (Pittsburgh National Bank Operations Building, Jenkins Arcade Bldg, Miscellaneous Bridges, and Federal Bldg) did not appear in any of the photographs, and were therefore not trained. Note also that the Blue Cross Bldg and the Grant Bldg were trained from test image 5 because they only appeared in that image.

	PPE Name	Symbol	Training Image
Rivers:	Allegheny River	AR	2
	Monongahela River	MR	7
	Ohio River	OR	4
Bridges:	Fort Duquesne Bridge	DB	4
	Fort Pitt Bridge	PB	4
	Ninth Ave. Bridge	NB	3
	Sixth Ave. Bridge	SB	2
	Stanwix St. Bridge Remnants	ST	7
	Miscellaneous Bridges	MB	
Roads:	Fort Duquesne Blvd.	DV	4
	Fort Pitt Blvd.	PV	4
	Miscellaneous Roads	R	7
Garages:	Eighth Ave. Parking	EP	2
	Ninth Ave. Parking Garage	NP	3
	Sixth Ave. Parking Garage	SP	2
Hotels:	Pick Roosevelt Hotel	PI	2
	Pittsburgh Hilton Hotel	HI	4
Banks:	Equibank Bldg.	EQ	2
	Mellon National Bank Bldg.	ME	7
	Pittsburgh National Bank Bldg.	PN	7
	Pittsburgh National Bank Operations Bldg.	PO	
Stores:	Gimbels Dept. Store	GI	3
	Jenkins Arcade Bldg.	JN	
	Joseph Hornes Dept. Store	HO	2
Government:	Federal Bldg.	FE	
	Pennsylvania State Office Bldg.	SO	6
	Pennsylvania State Office Bldg. Lobby	SL	6
Apartments:	Gateway Towers Apartments	GT	1
	Penthouse Apartments	PE	2

	PPE Name	Symbol	Training Image
Misc. Offices:	Alcoa Bldg.	AL	3
	Allegheny Towers Bldg.	AT	2
	Bell Telephone Co. Bldg.	BT	3
	Blue Cross Bldg.	BL	Test 5
	Fulton Bldg.	FU	2
	Gateway Center Bldg. 1	GA	2
	Gateway Center Bldg. 2	GB	2
	Gateway Center Bldg. 3	GC	1
	Gateway Center Bldg. 4	GD	6
	Grant Bldg.	GR	Test 5
	Gulf Bldg.	GU	3
	I. B. M. Bldg.	IB	6
	Koppers Bldg.	KO	3
	Miscellaneous Buildings	B	7
	Oliver Bldg.	OL	4
	One Oliver Plaza	OO	2
	Penn Technical Center	PT	3
	Pittsburgh Press	PR	4
	Rust Bldg.	RU	3
	Shields Rubber Bldg.	SH	7
	U. S. Steel Bldg.	US	4
	United Engineering & Foundry Bldg.	UE	3
	Westinghouse Bldg.	WS	7
	Westinghouse Plaza	WP	7
Miscellaneous:	Mountains	M	4
	Park	P	6
	Sky	SK	7
	Snow	SN	7
	Three Rivers Stadium	TR	5

APPENDIX II: Primitive Picture Elements (continued)

This table shows all of the primitive picture elements (PPEs) that were selected for the view angle identification task of the Pittsburgh images. The symbol column is the identifier used in the labelings in Appendices VI, VII, and VIII.

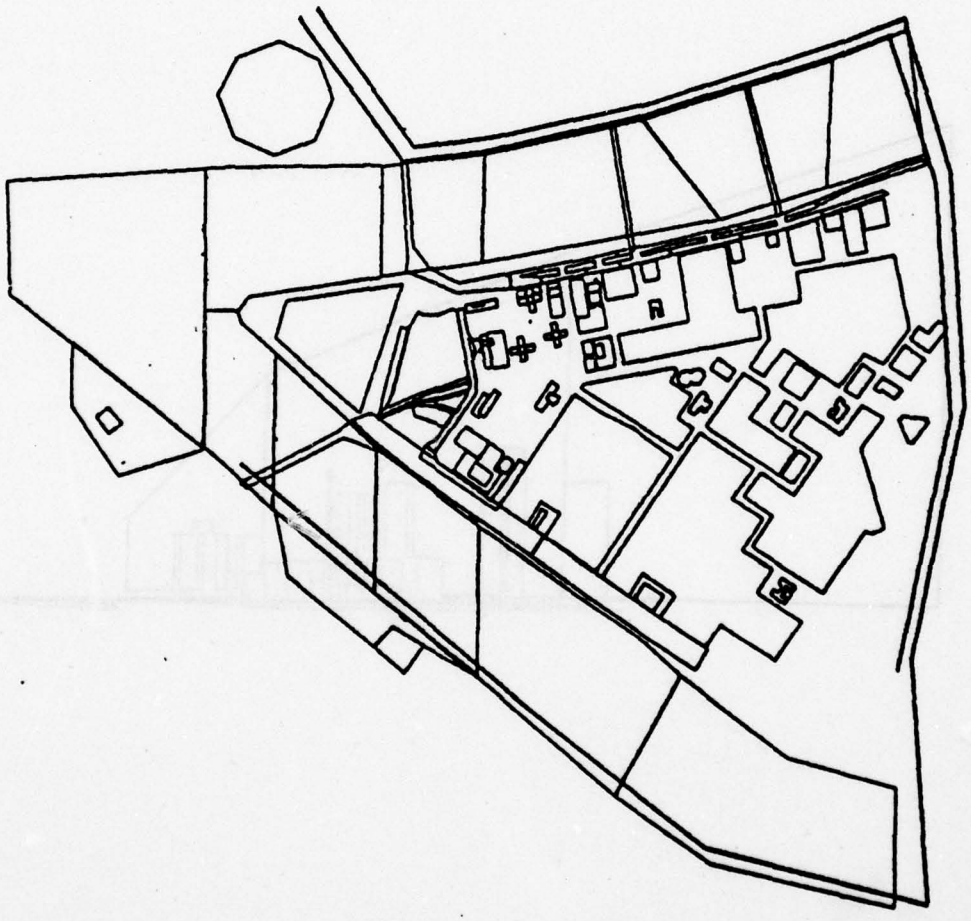
PPE Name	Symbol
Allegheny River	AR
Grant Bldg.	GR
Gulf Bldg.	GU
Mellon National Bank Bldg.	ME
Miscellaneous Bridges	MB
Miscellaneous Buildings	B
Miscellaneous Roads	R
Monongahela River	MR
Mountains	M
Ohio River	OR
One Oliver Plaza	OO
Park	P
Sky	SK
Three Rivers Stadium	TR
U. S. Steel Bldg.	US
Westinghouse Bldg.	WS

APPENDIX III: Original City Map

Source: City of Pittsburgh Planning Office

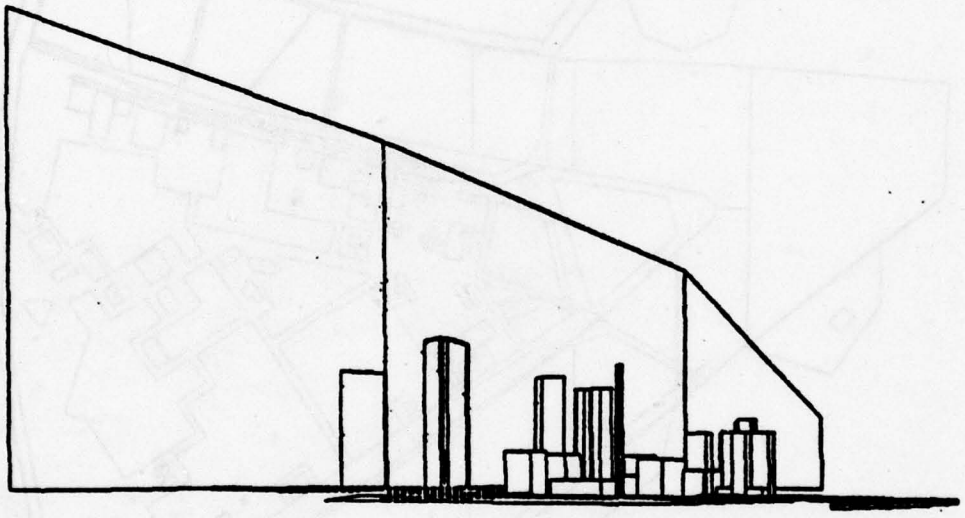
APPENDIX IV: DIGITIZED CITY MAP

This is the city of Pittsburgh as ARGOS sees it. Notice that the rivers and roads are segmented into small pieces. This is because the map is 3" by 2" and had to be broken down into more manageable pieces before digitization. Notice also that the sky and mountain regions are represented as lines that surround the city limits. When hypothesized views are generated, these lines form "backdrops" that correctly represent the regions. The large regions in the center of the city are miscellaneous buildings.



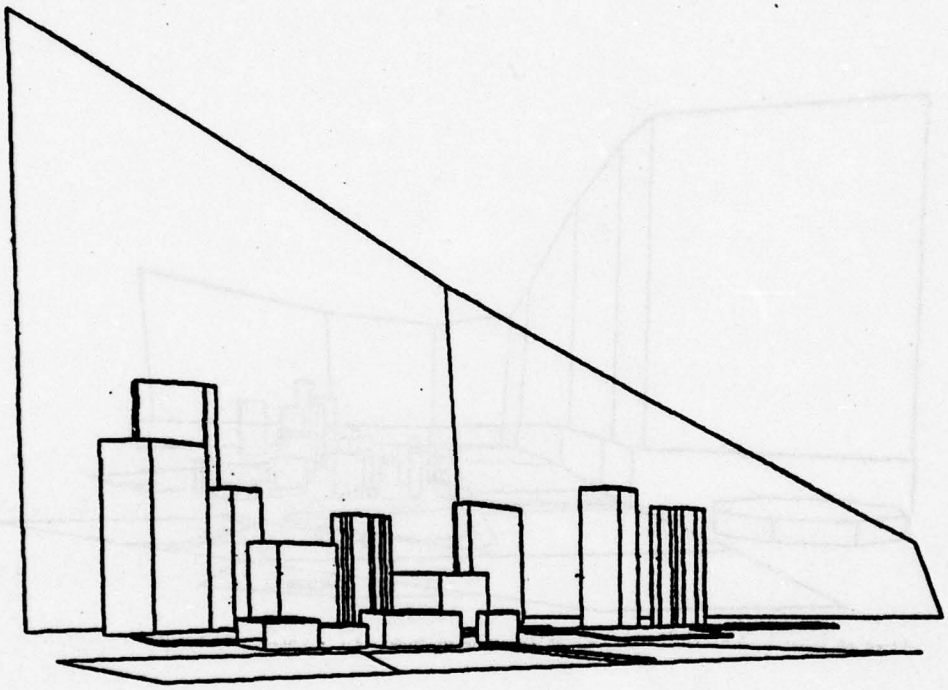
APPENDIX V: GENERATED VIEWS

This is the view of the internal model that was generated to simulate training scene 2. It was taken from a declination of 1^{0} (1^{0} above the horizon) and a right-ascension of 300^{0} (where right-ascension of 0^{0} is due west and proceeds counter-clockwise). The large three-piece region in the background is the sky.



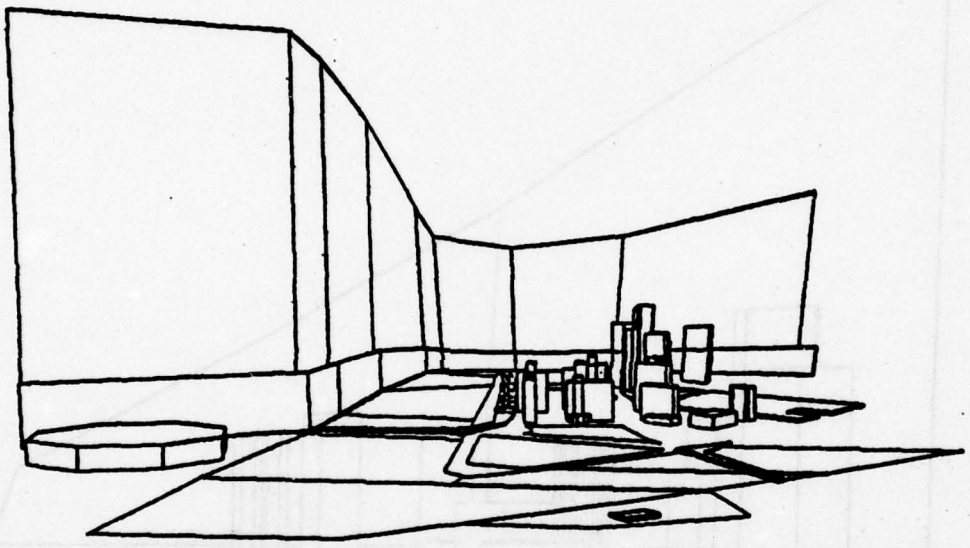
APPENDIX V: GENERATED VIEWS (continued)

This is the view of the internal model that was generated to simulate training scene 3. It was taken from a declination of 2^{o} and a right-ascension of 245^{o}.



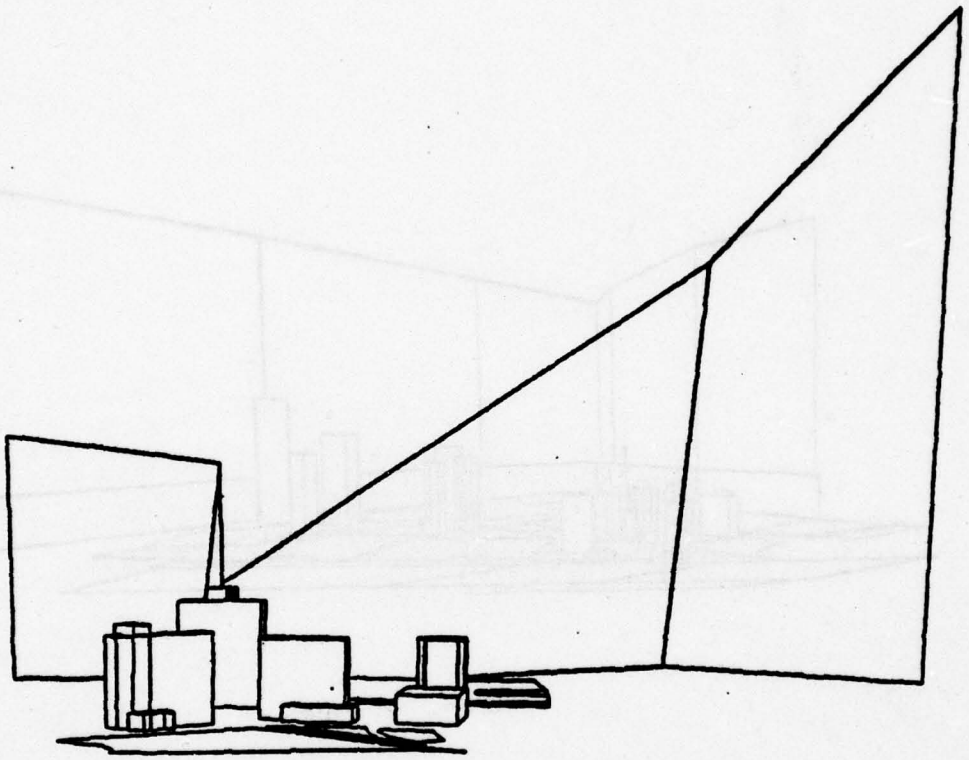
APPENDIX V: GENERATED VIEWS (continued)

This is the view of the internal model that was generated to simulate training scene 5. It was taken from a declination of 5^{0} and a right-ascension of 10^{0}.



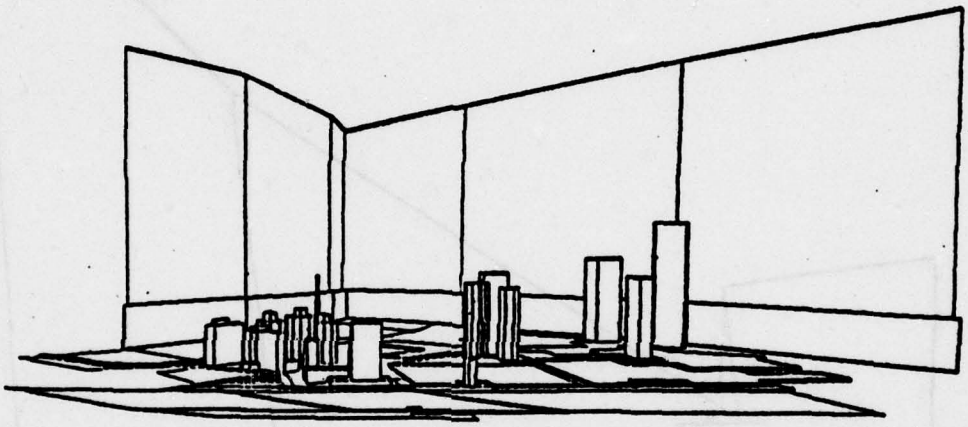
APPENDIX V: GENERATED VIEWS (continued)

This is the view of the internal model that was generated to simulate training scene 6. It was taken from a declination of 5^{o} and a right-ascension of 345^{o}.



APPENDIX V: GENERATED VIEWS (continued)

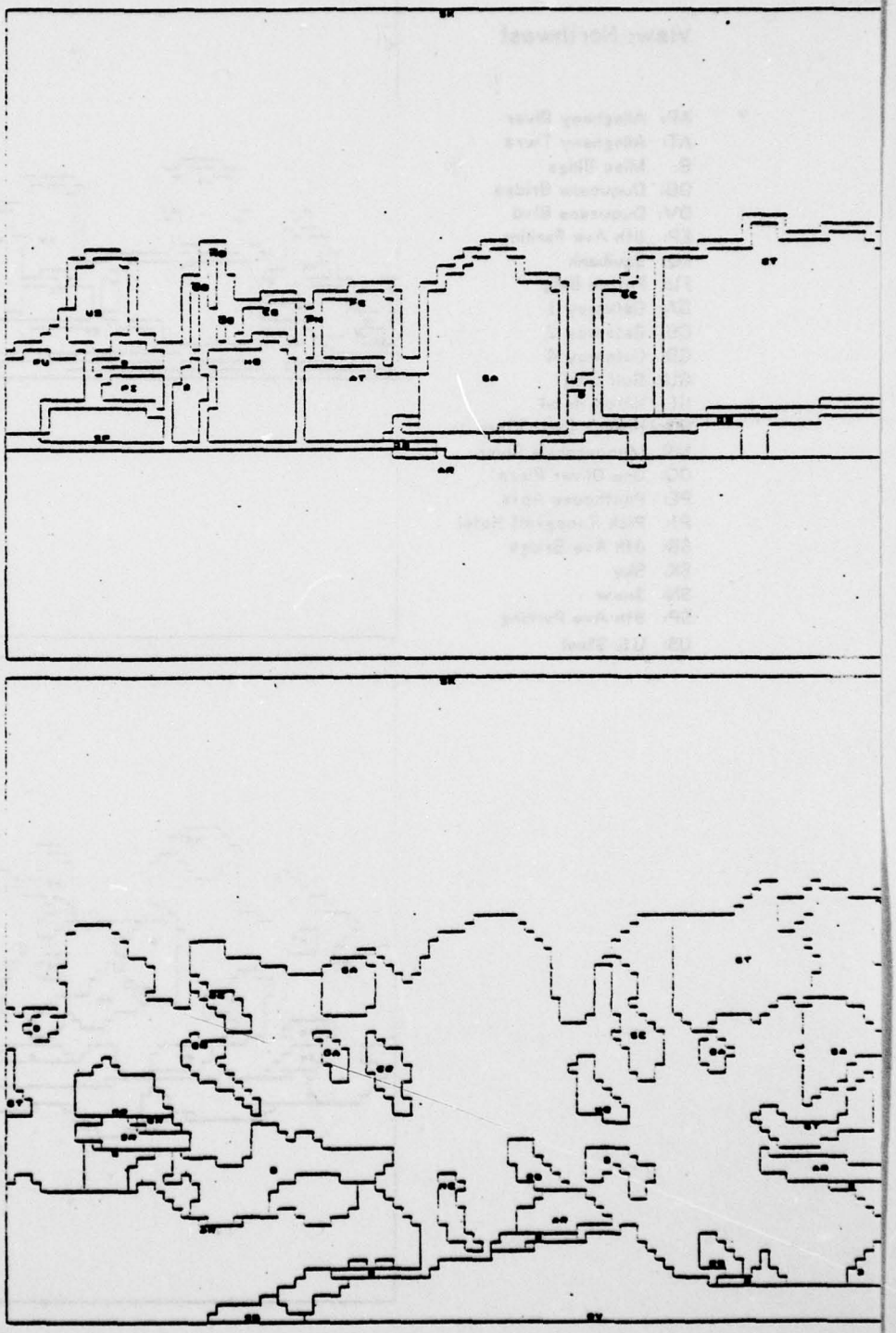
This is the view of the internal model that was generated to simulate training scene 7. It was taken from a declination of 3^{o} and a right-ascension of 55^{o}.



APPENDIX VI: Human and ARGOS Labeling, Training Scene 1

View: Northwest

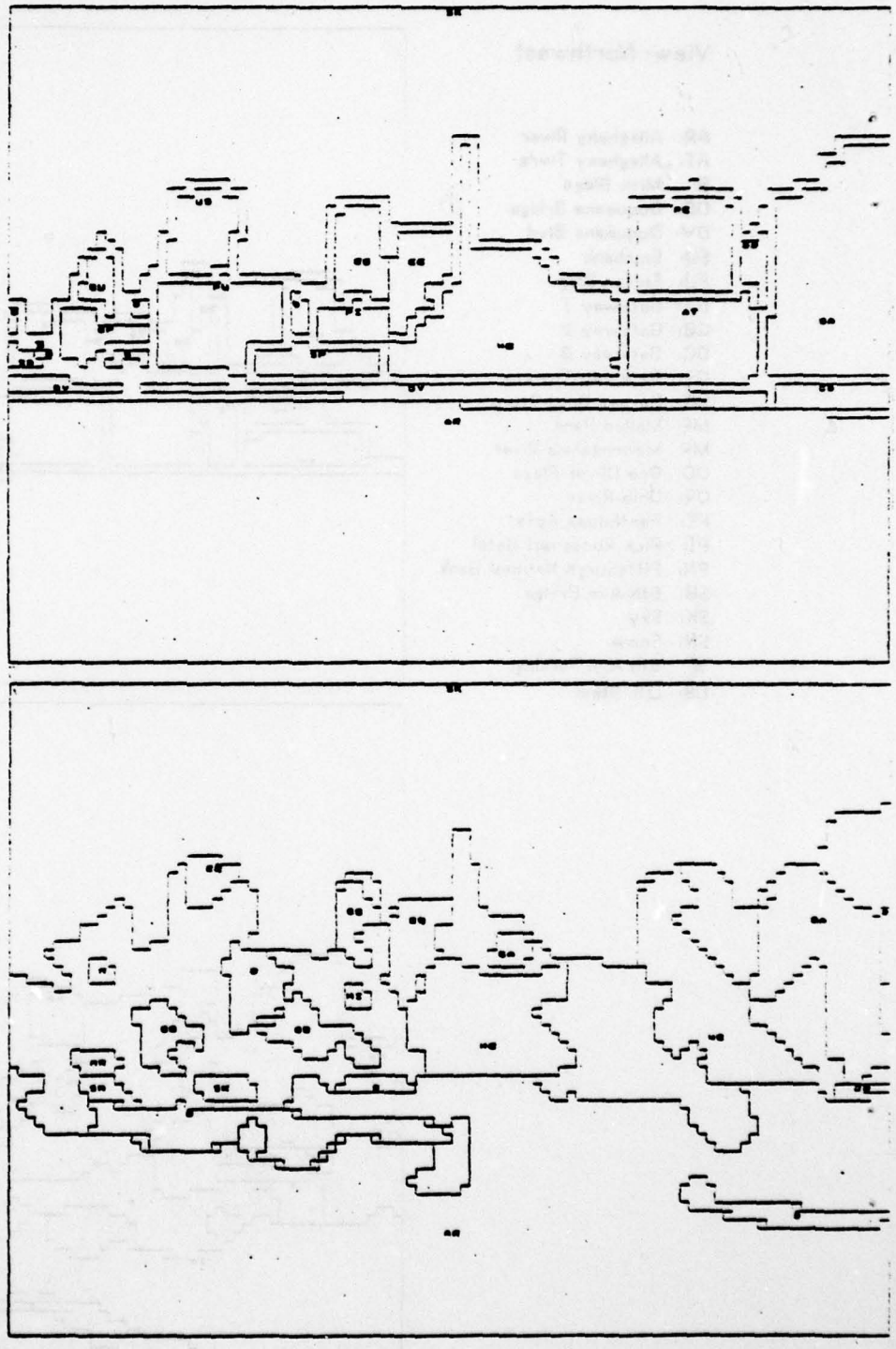
AR: Allegheny River
AT: Allegheny Twrs
B: Misc Bldgs
DB: Duquesne Bridge
DV: Duquesne Blvd
EQ: Equibank
FU: Fulton Bldg
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GT: Gateway Towers
HO: Hornes Dept Store
ME: Mellon Bank
MR: Monongahela River
OO: One Oliver Plaza
OR: Ohio River
PE: Penthouse Apts
PI: Pick Roosevelt Hotel
PN: Pittsburgh National Bank
SB: 6th Ave Bridge
SK: Sky
SN: Snow
SP: 6th Ave Parking
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Training Scene 2

View: Northwest

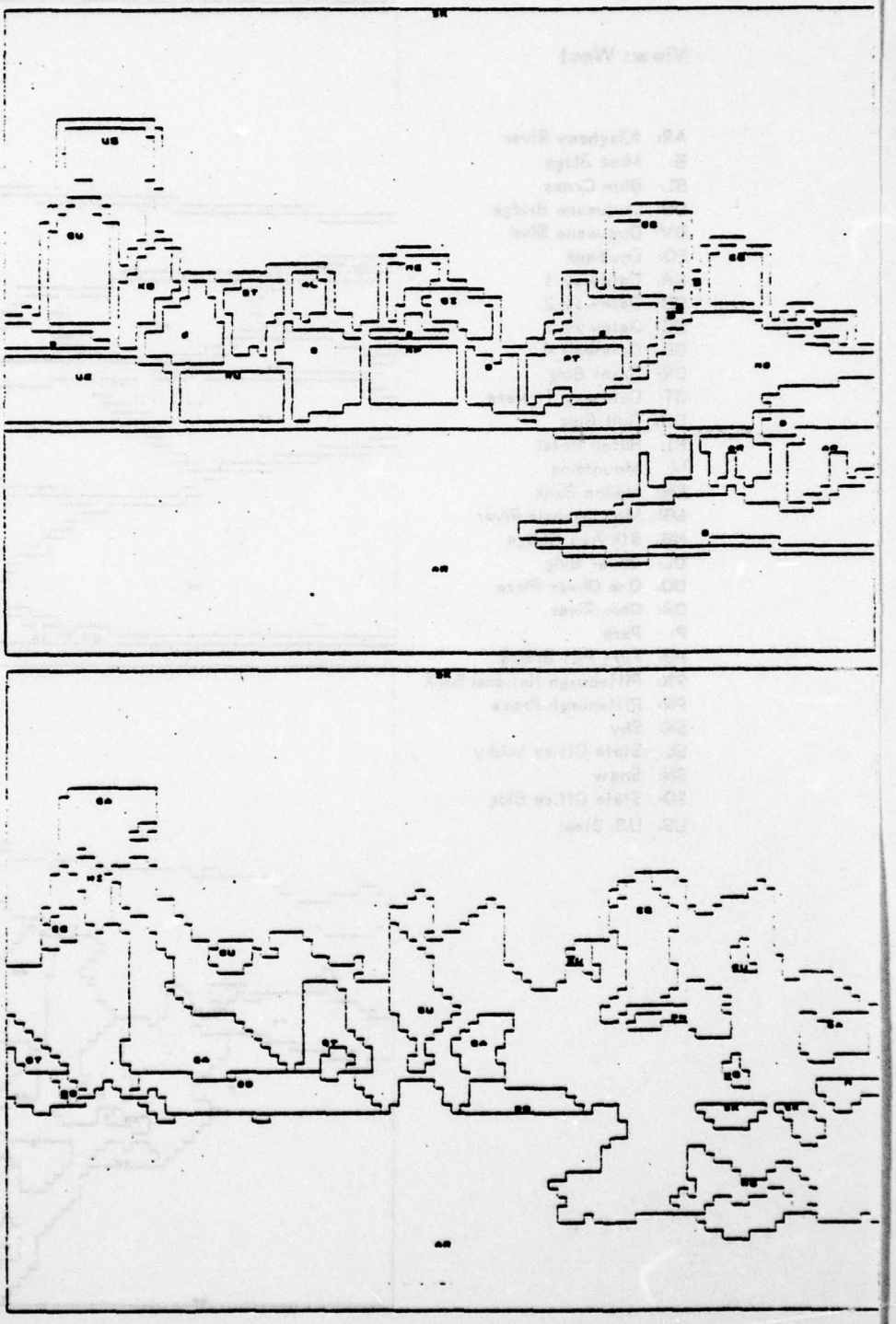
AR: Allegheny River
AT: Allegheny Twrs
B: Misc Bldgs
DB: Duquesne Bridge
DV: Duquesne Blvd
EP: 8th Ave Parking
EQ: Equibank
FU: Fulton Bldg
GA: Gateway 1
GB: Gateway 2
GD: Gateway 4
GU: Gulf Bldg
HI: Hilton Hotel
HO: Hornes Dept Store
MR: Monongahela River
OO: One Oliver Plaza
PE: Penthouse Apts
PI: Pick Roosevelt Hotel
SB: 6th Ave Bridge
SK: Sky
SN: Snow
SP: 6th Ave Parking
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Training Scene 3

View: Northeast

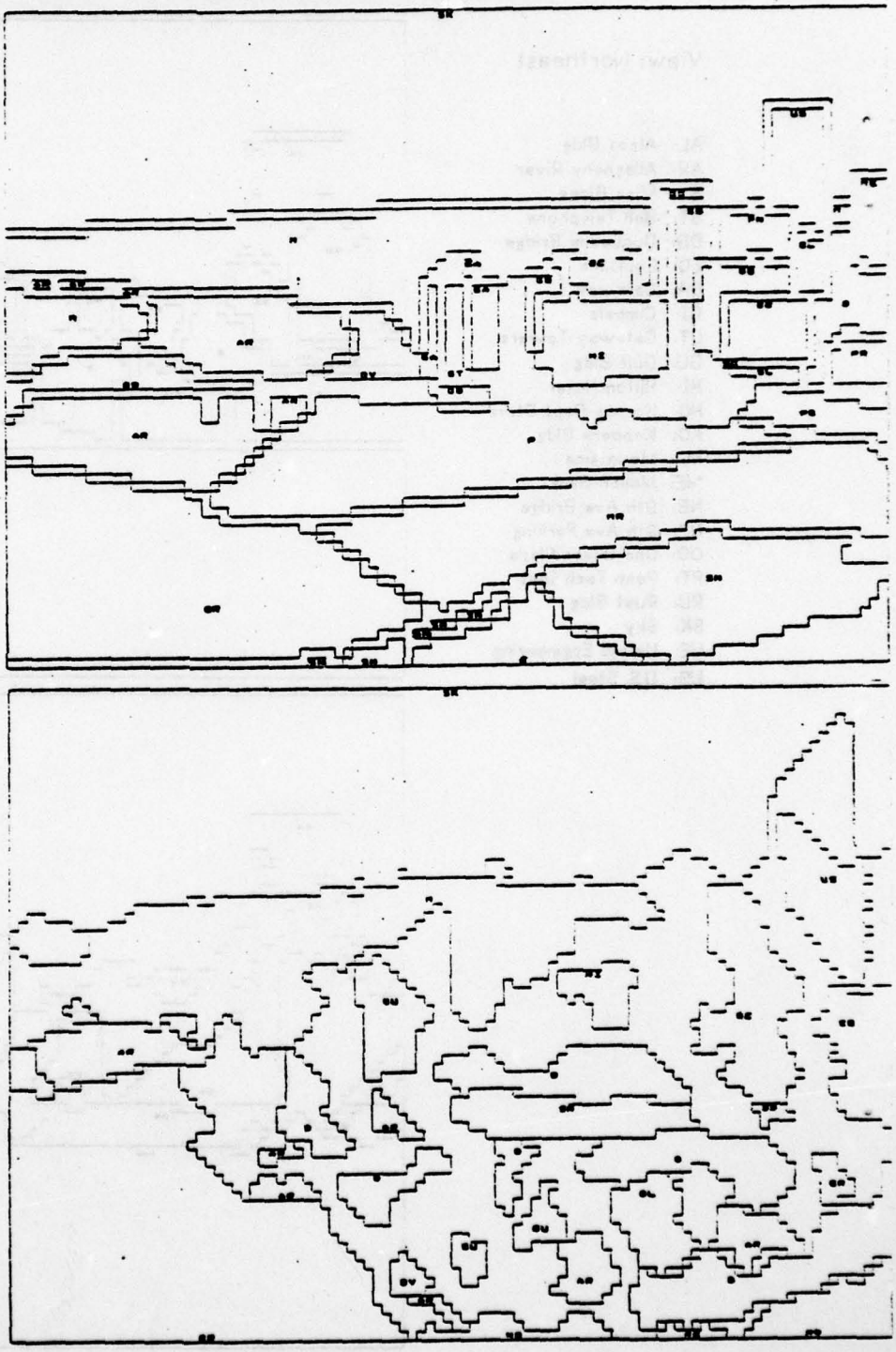
AL: Alcoa Bldg
AR: Allegheny River
B: Misc Bldgs
BT: Bell Telephone
DB: Duquesne Bridge
EQ: Equibank
GA: Gateway 1
GI: Gimbels
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
HO: Hornes Dept Store
KO: Koppers Bldg
M: Mountains
ME: Mellon Bank
NB: 9th Ave Bridge
NP: 9th Ave Parking
OO: One Oliver Plaza
PT: Penn Tech Inst
RU: Rust Bldg
SK: Sky
UE: United Engineering
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Training Scene 4

View: West

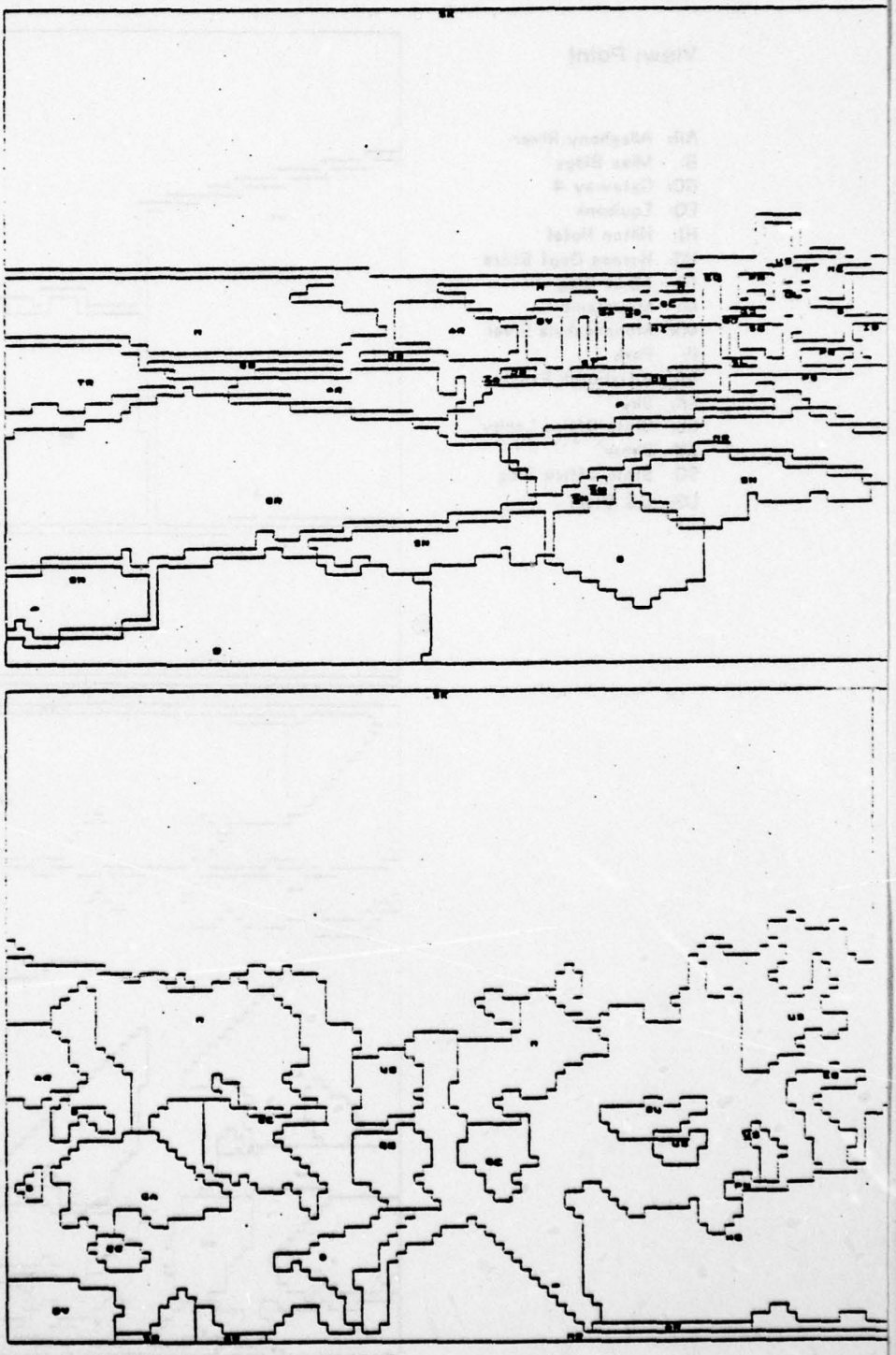
AR: Allegheny River
B: Misc Bldgs
BL: Blue Cross
DB: Duquesne Bridge
DV: Duquesne Blvd
EQ: Equibank
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GD: Gateway 4
GR: Grant Bldg
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
M: Mountains
ME: Mellon Bank
MR: Monongahela River
NB: 9th Ave Bridge
OL: Oliver Bldg
OO: One Oliver Plaza
OR: Ohio River
P: Park
PB: Fort Pitt Bridge
PN: Pittsburgh National Bank
PR: Pittsburgh Press
SK: Sky
SL: State Office Lobby
SN: Snow
SO: State Office Bldg
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Training Scene 5

View: West

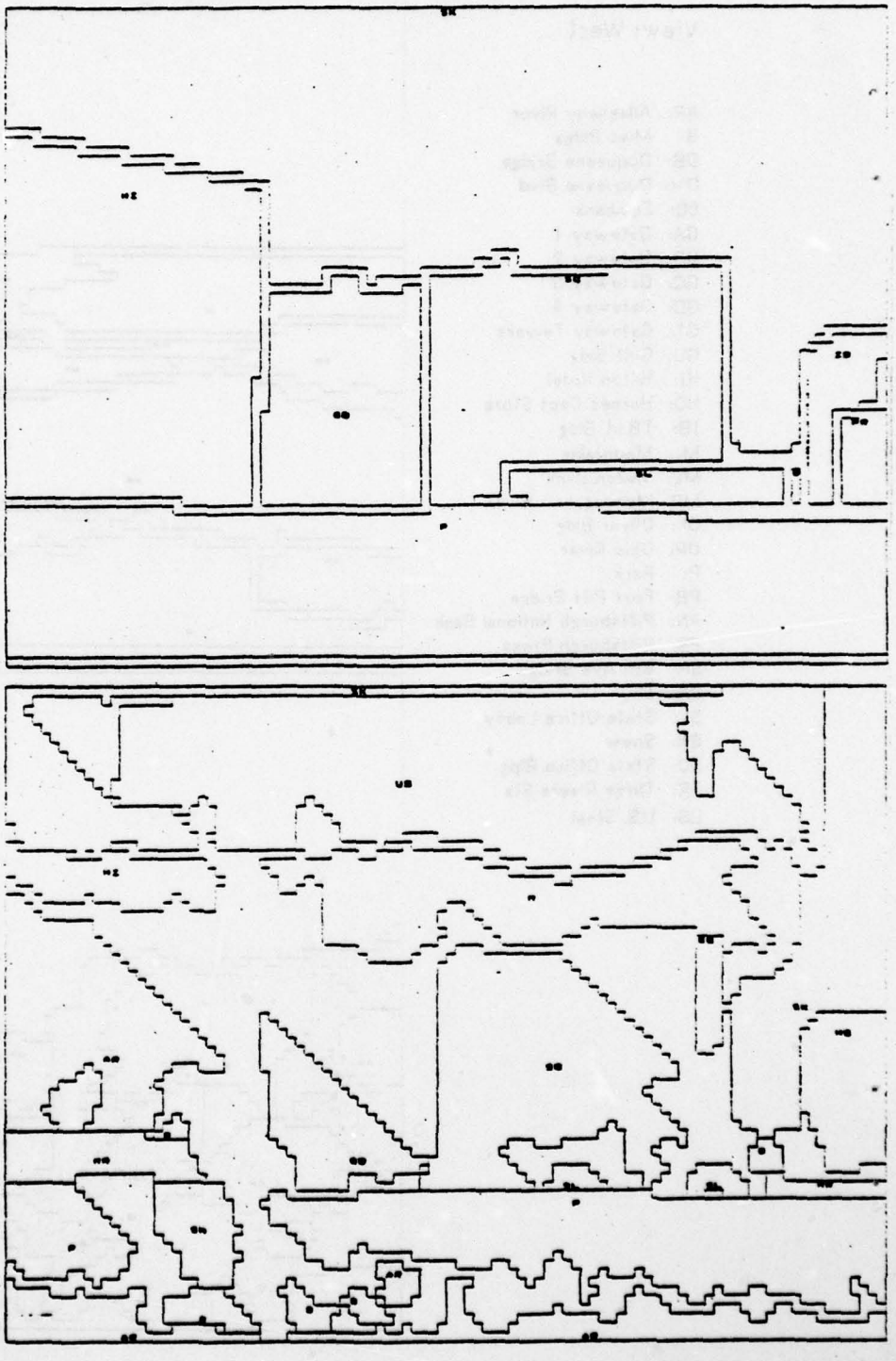
AR: Allegheny River
B: Misc Bldgs
DB: Duquesne Bridge
DV: Duquesne Blvd
EQ: Equibank
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GD: Gateway 4
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
HO: Hornes Dept Store
IB: I.B.M. Bldg
M: Mountains
ME: Mellon Bank
MR: Monongahela River
OL: Oliver Bldg
OR: Ohio River
P: Park
PB: Fort Pitt Bridge
PN: Pittsburgh National Bank
PR: Pittsburgh Press
SB: 6th Ave Bridge
SK: Sky
SL: State Office Lobby
SN: Snow
SO: State Office Bldg
TR: Three Rivers Sta
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Training Scene 6

View: Point

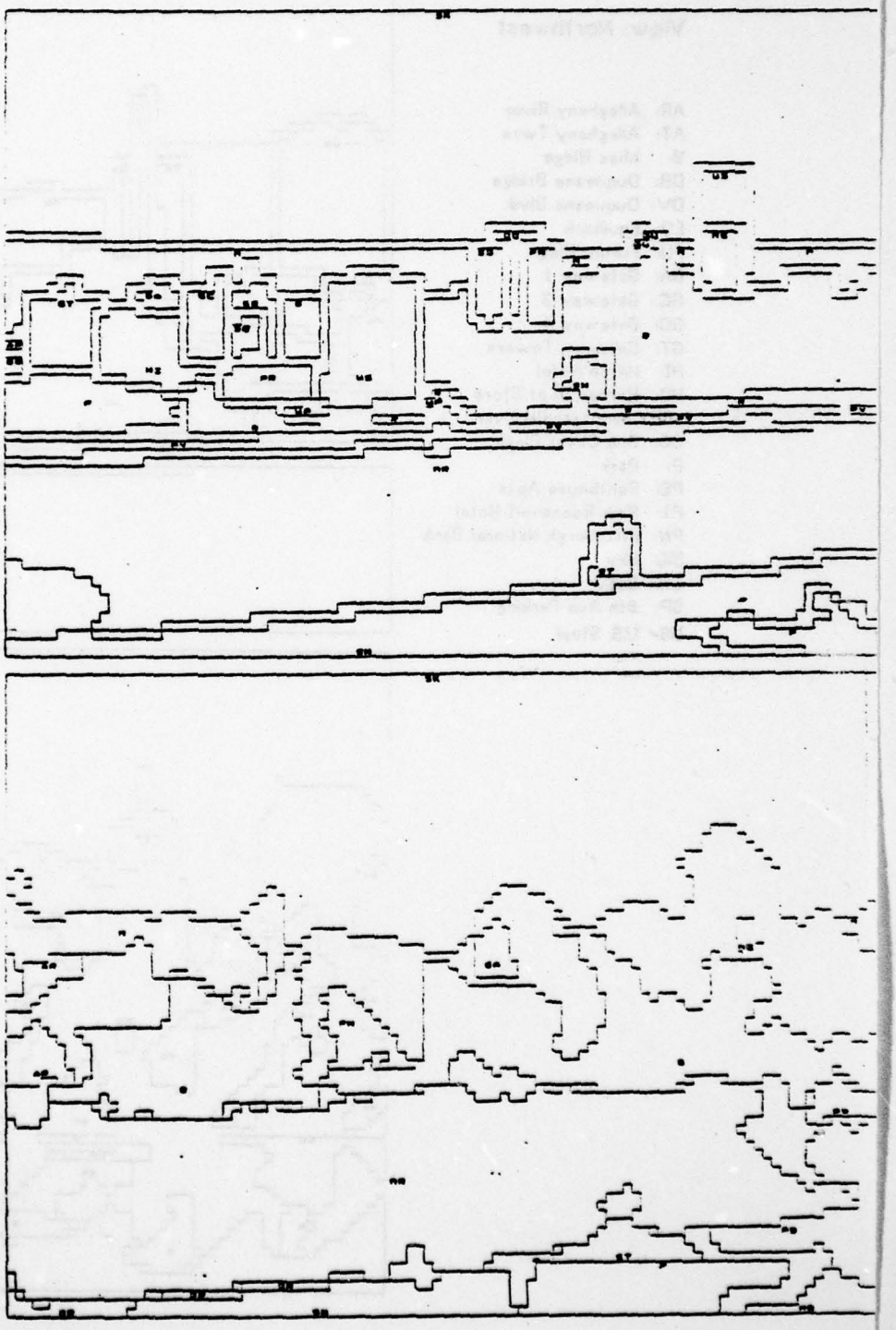
AR: Allegheny River
B: Misc Bldgs
GD: Gateway 4
EQ: Equibank
HI: Hilton Hotel
HO: Hornes Dept Store
IB: I.B.M. Bldg
M: Mountains
MR: Monongahela River
P: Park
PR: Pittsburgh Press
SK: Sky
SL: State Office Lobby
SN: Snow
SO: State Office Bldg
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Training Scene 7

View: Southwest

AR: Allegheny River
B: Misc Bldgs
DB: Duquesne Bridge
EQ: Equibank
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
M: Mountains
ME: Mellon Bank
MR: Monongahela River
OO: One Oliver Plaza
OR: Ohio River
P: Park
PB: Fort Pitt Bridge
PN: Pittsburgh National Bank
PR: Pittsburgh Press
PV: Fort Pitt Blvd
R: Misc Roads
SH: Shields Rubber Bldg
SK: Sky
SN: Snow
SO: State Office Bldg
ST: Stanwix Bridge
US: U.S. Steel
WP: Westinghouse Plaza
WS: Westinghouse Bldg



AD-A066 736 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER --ETC F/G 9/1
THE ARGOS IMAGE UNDERSTANDING SYSTEM,(U)
NOV 78 S RUBIN F44620-73-C-0074

UNCLASSIFIED AFOSR-TR-79-0087 NL

2 OF 2
ADA
066736

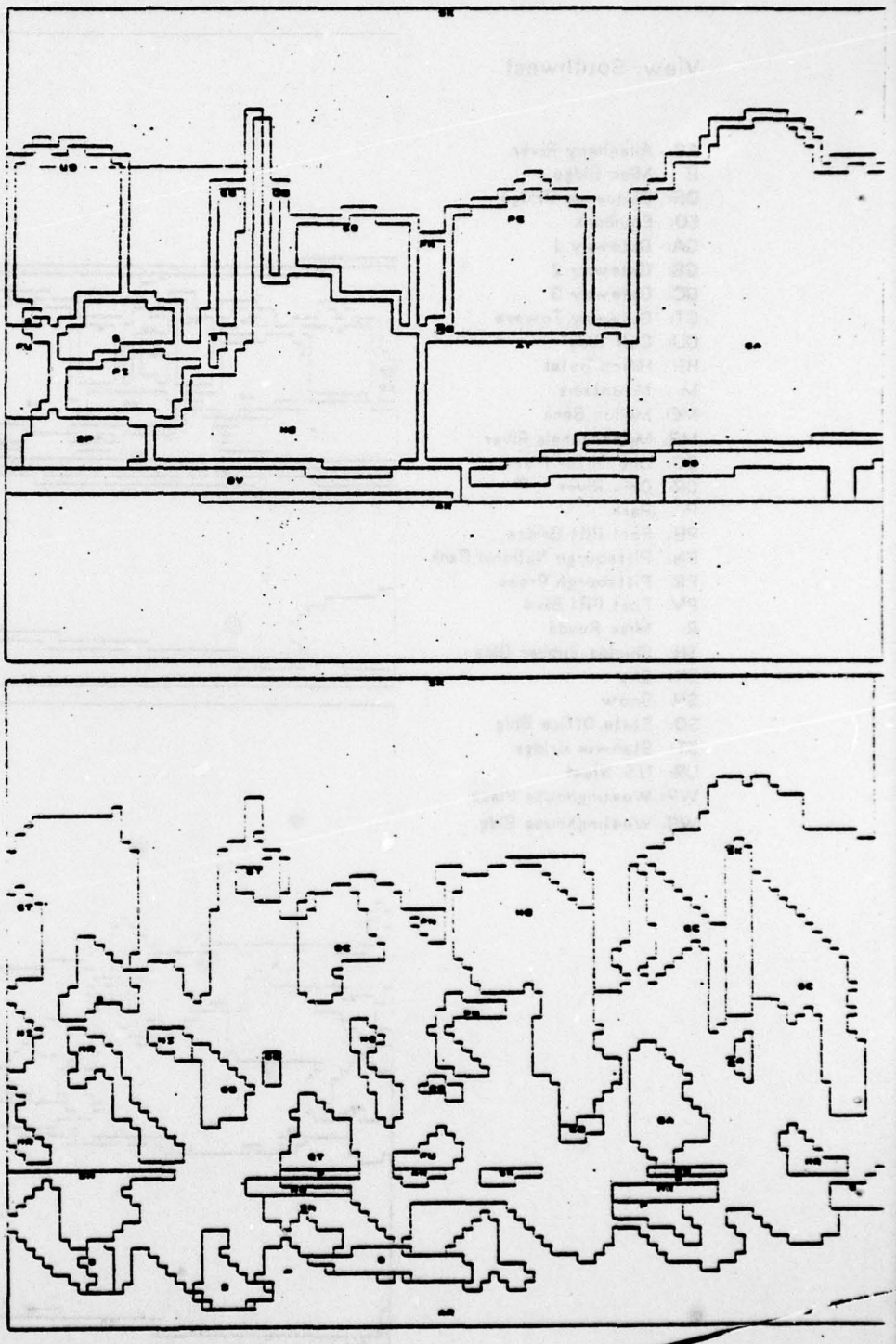


END
DATE
FILMED
5-79
DDC

APPENDIX VI: Human and ARGOS Labeling, Test Scene 1

View: Northwest

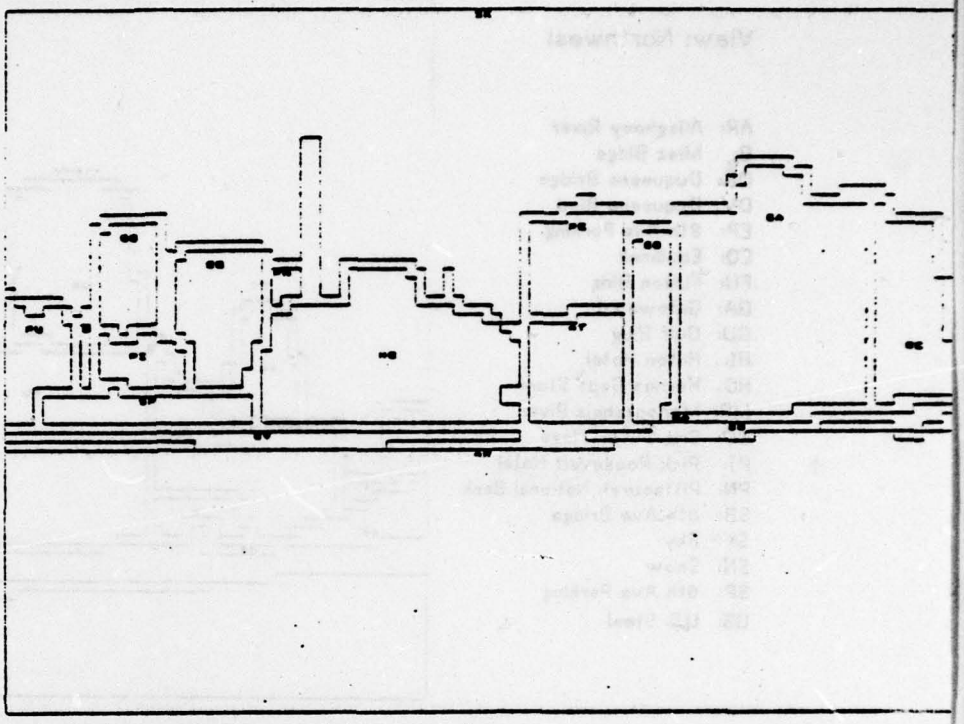
AR: Allegheny River
AT: Allegheny Twrs
B: Misc Bldgs
DB: Duquesne Bridge
DV: Duquesne Blvd
EQ: Equibank
FU: Fulton Bldg
GA: Gateway 1
GC: Gateway 3
GD: Gateway 4
GT: Gateway Towers
HI: Hilton Hotel
HO: Hornes Dept Store
MR: Monongahela River
OO: One Oliver Plaza
P: Park
PE: Penthouse Apts
PI: Pick Roosevelt Hotel
PN: Pittsburgh National Bank
SK: Sky
SN: Snow
SP: 6th Ave Parking
US: U.S. Steel

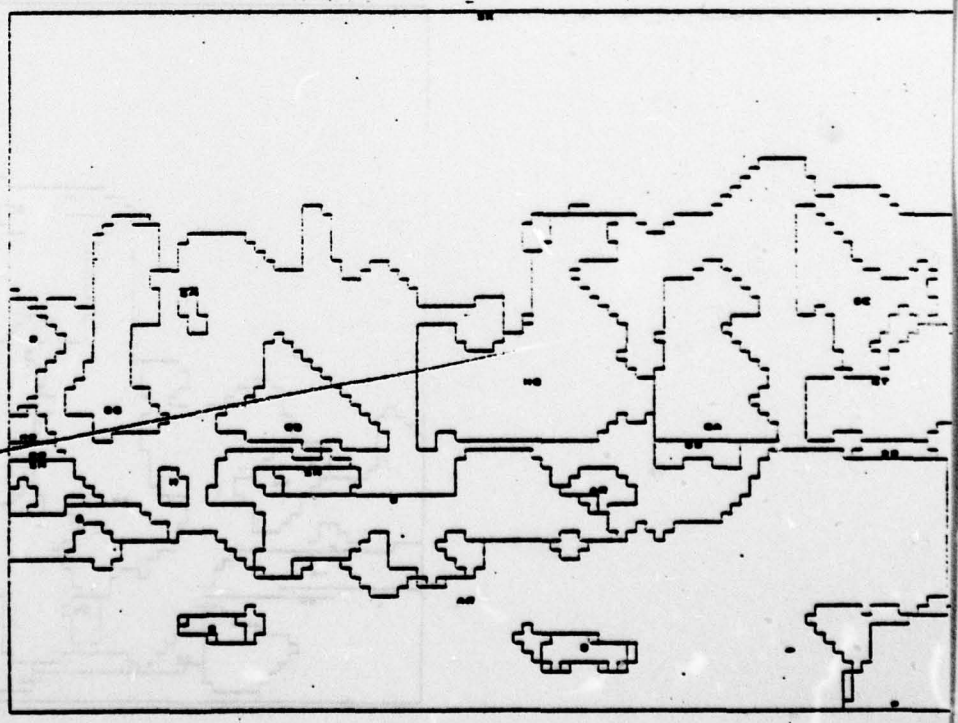


APPENDIX VI: Human and ARGOS Labeling, Test Scene 2

View: Northwest

AR: Allegheny River
AT: Allegheny Twrs
B: Misc Bldgs
DB: Duquesne Bridge
DV: Duquesne Blvd
EQ: Equibank
FU: Fulton Bldg
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GD: Gateway 4
GT: Gateway Towers
HO: Hornes Dept Store
M: Mountains
MR: Monongahela River
OO: One Oliver Plaza
OR: Ohio River
P: Park
PE: Penthouse Apts
PI: Pick Roosevelt Hotel
PN: Pittsburgh National Bank
SK: Sky
SN: Snow
SP: 6th Ave Parking

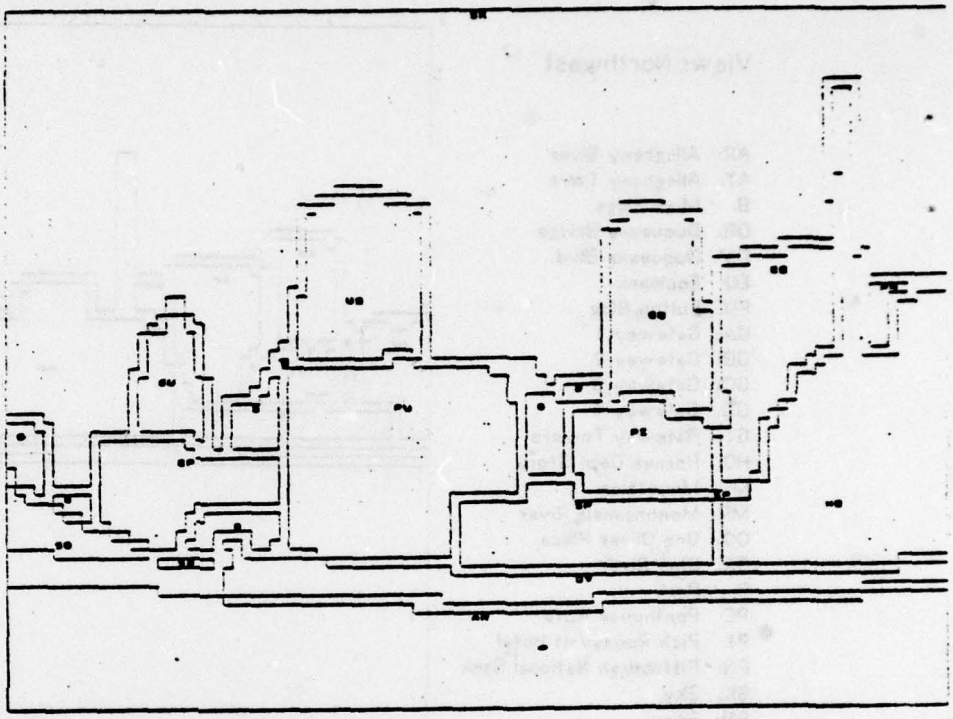


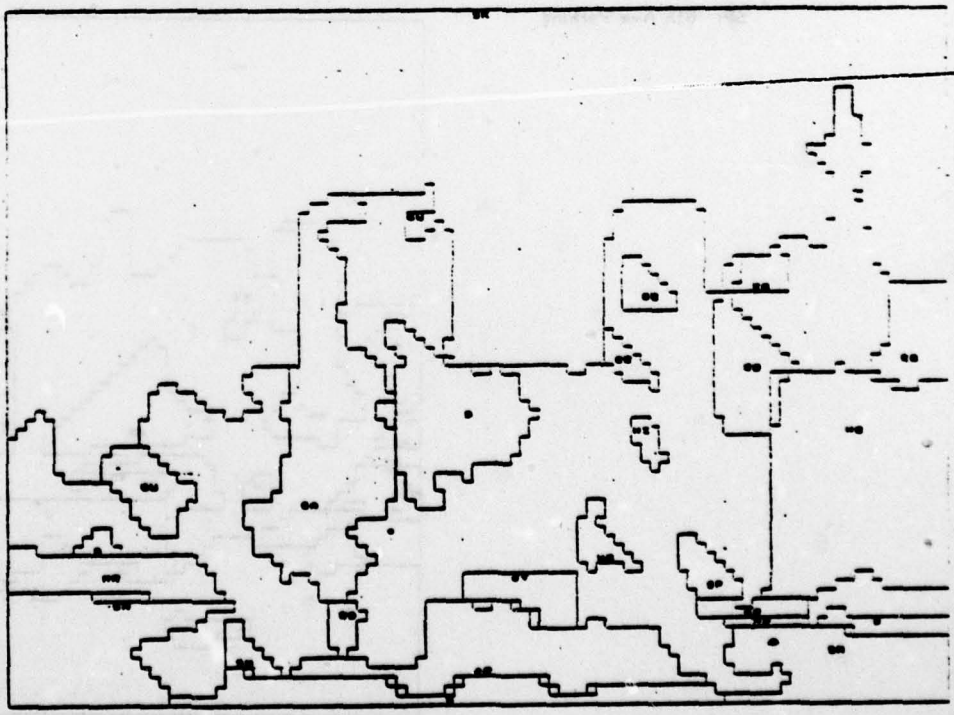


APPENDIX VI: Human and ARGOS Labeling, Test Scene 3

View: Northwest

AR: Allegheny River
B: Misc Bldgs
DB: Duquesne Bridge
DV: Duquesne Blvd
EP: 8th Ave Parking
EQ: Equibank
FU: Fulton Bldg
GA: Gateway 1
GU: Gulf Bldg
HI: Hilton Hotel
HO: Hornes Dept Store
MR: Monongahela River
OO: One Oliver Plaza
PI: Pick Roosevelt Hotel
PN: Pittsburgh National Bank
SB: 6th Ave Bridge
SK: Sky
SN: Snow
SP: 6th Ave Parking
US: U.S. Steel

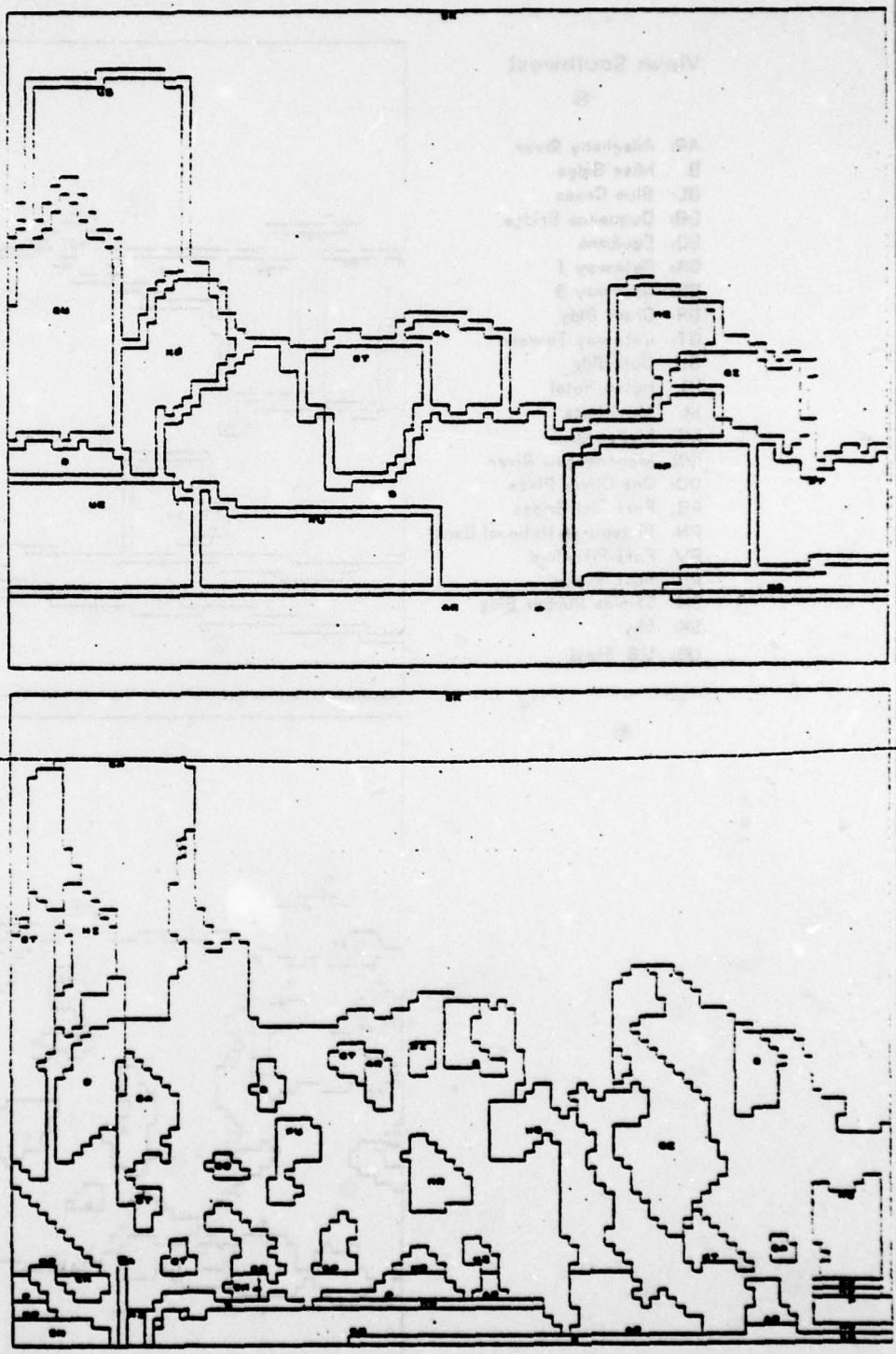




APPENDIX VI: Human and ARGOS Labeling, Test Scene 4

View: Northeast

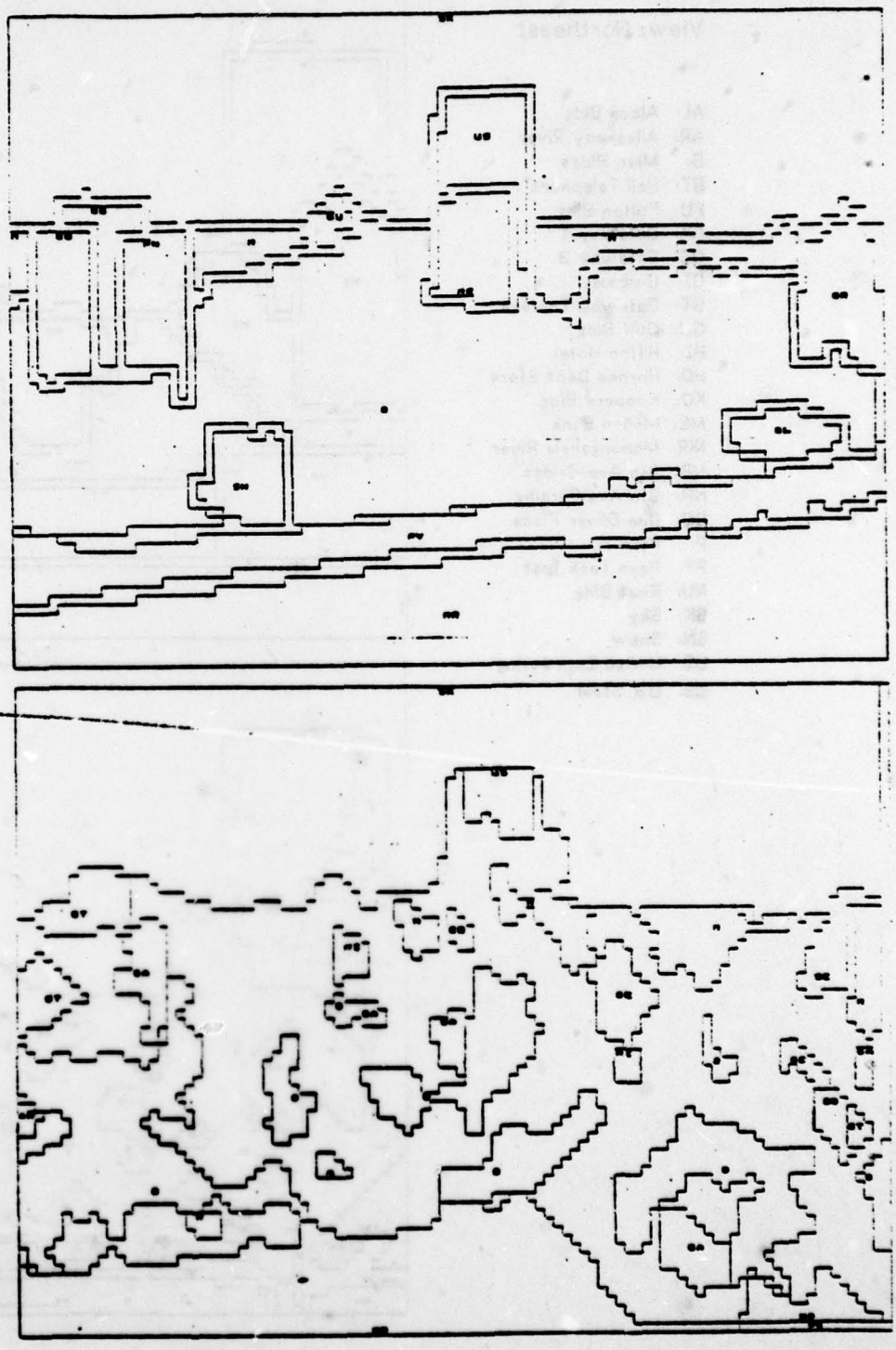
AL: Alcoa Bldg
AR: Allegheny River
B: Misc Bldgs
BT: Bell Telephone
FU: Fulton Bldg
GA: Gateway 1
GC: Gateway 3
GI: Gimbels
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
HO: Hornes Dept Store
KO: Koppers Bldg
ME: Mellon Bank
MR: Monongahela River
NB: 9th Ave Bridge
NP: 9th Ave Parking
OO: One Oliver Plaza
P: Park
PT: Penn Tech Inst
RU: Rust Bldg
SK: Sky
SN: Snow
UE: United Engineering
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Test Scene 5

View: Southwest

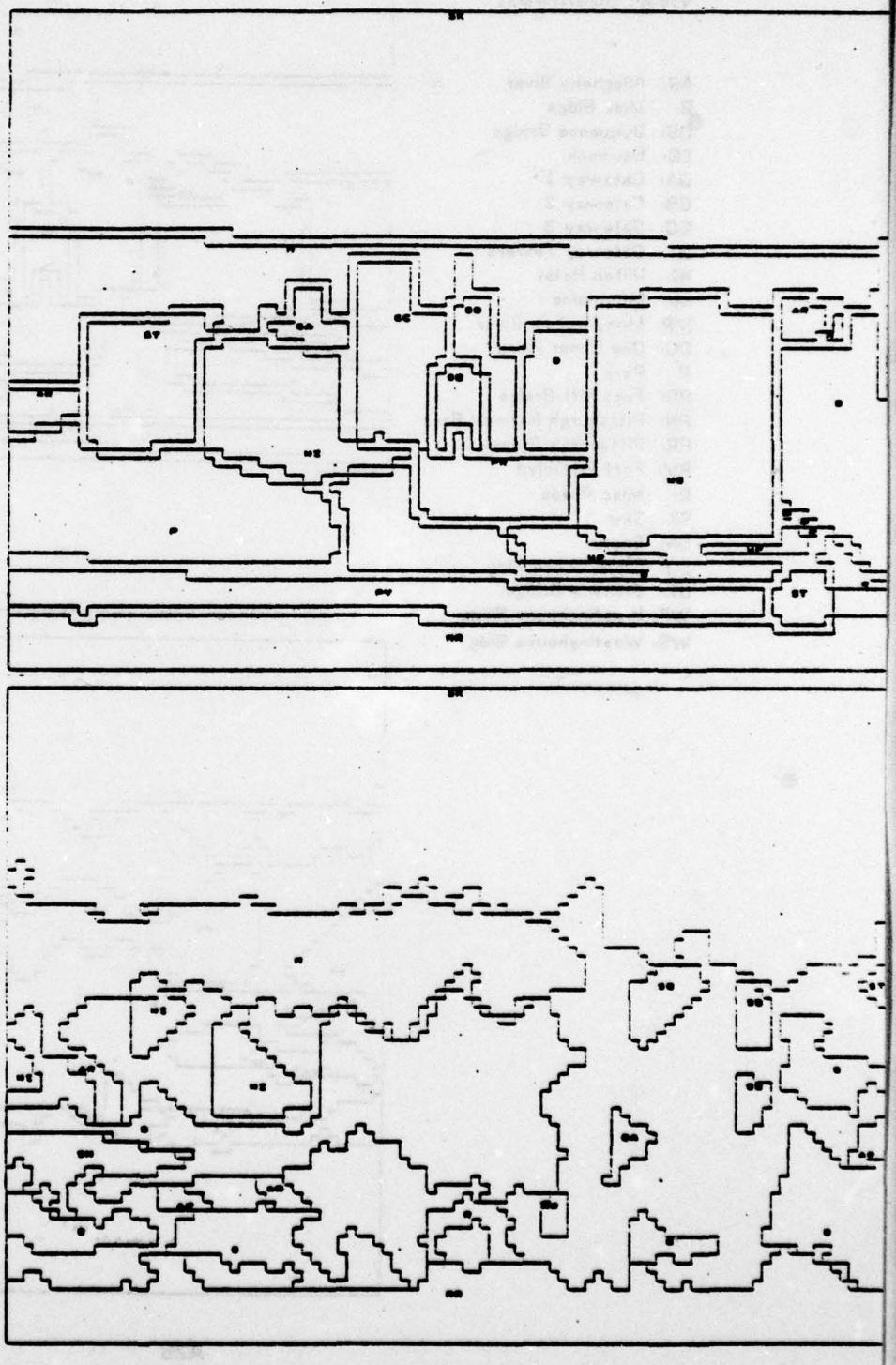
AR: Allegheny River
B: Misc Bldgs
BL: Blue Cross
DB: Duquesne Bridge
EQ: Equibank
GA: Gateway 1
GC: Gateway 3
GR: Grant Bldg
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
M: Mountains
ME: Mellon Bank
MR: Monongahela River
OO: One Oliver Plaza
PB: Fort Pitt Bridge
PN: Pittsburgh National Bank
PV: Fort Pitt Blvd
R: Misc Roads
SH: Shields Rubber Bldg
SK: Sky
US: U.S. Steel



APPENDIX VI: Human and ARGOS Labeling, Test Scene 6

View: Southwest

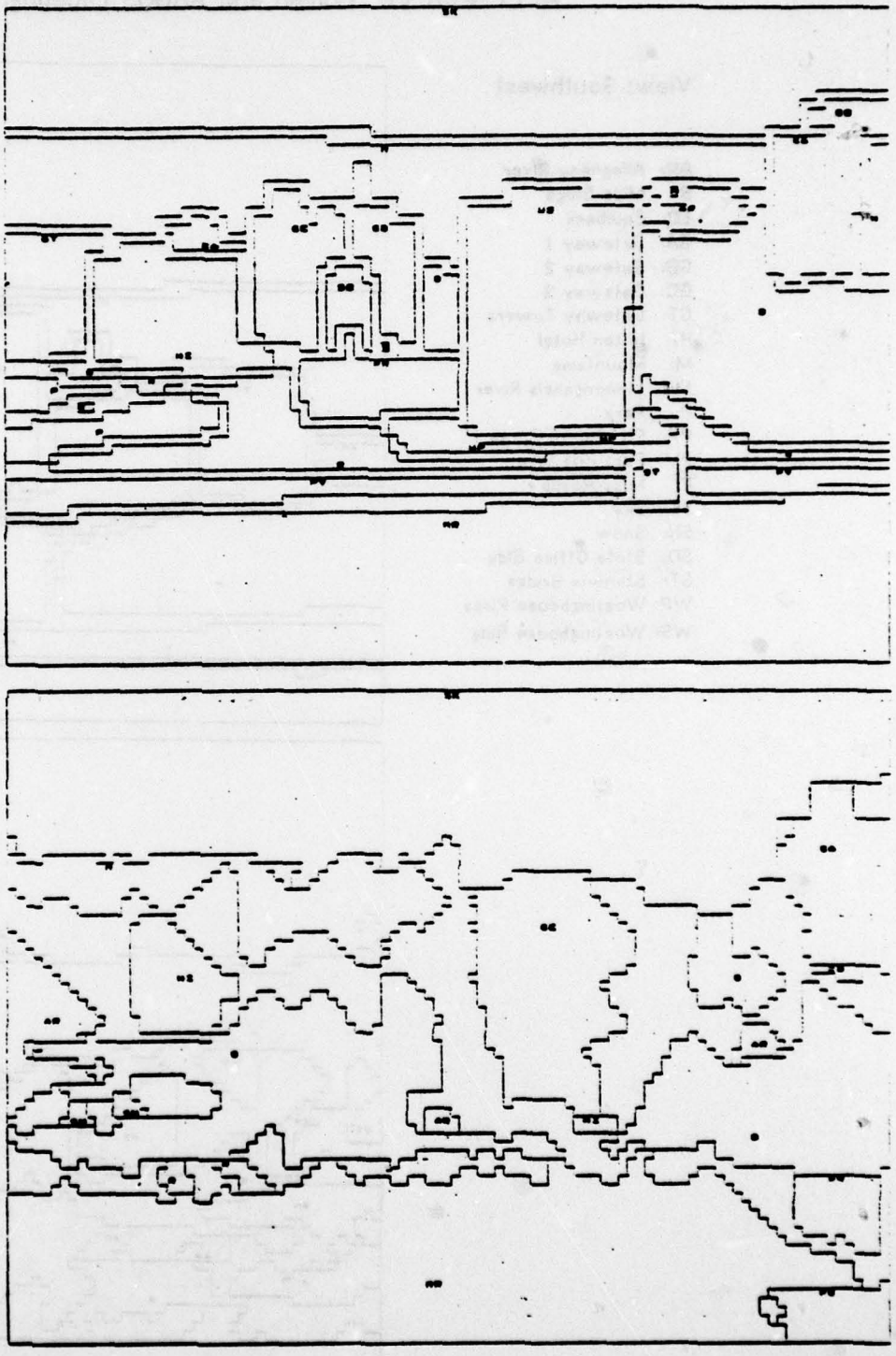
AR: Allegheny River
B: Misc Bldgs
EQ: Equibank
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GT: Gateway Towers
HI: Hilton Hotel
M: Mountains
MR: Monongahela River
P: Park
PR: Pittsburgh Press
PV: Fort Pitt Blvd
R: Misc Roads
SK: Sky
SN: Snow
SO: State Office Bldg
ST: Stanwix Bridge
WP: Westinghouse Plaza
WS: Westinghouse Bldg



APPENDIX VI: Human and ARGOS Labeling, Test Scene 7

View: Southwest

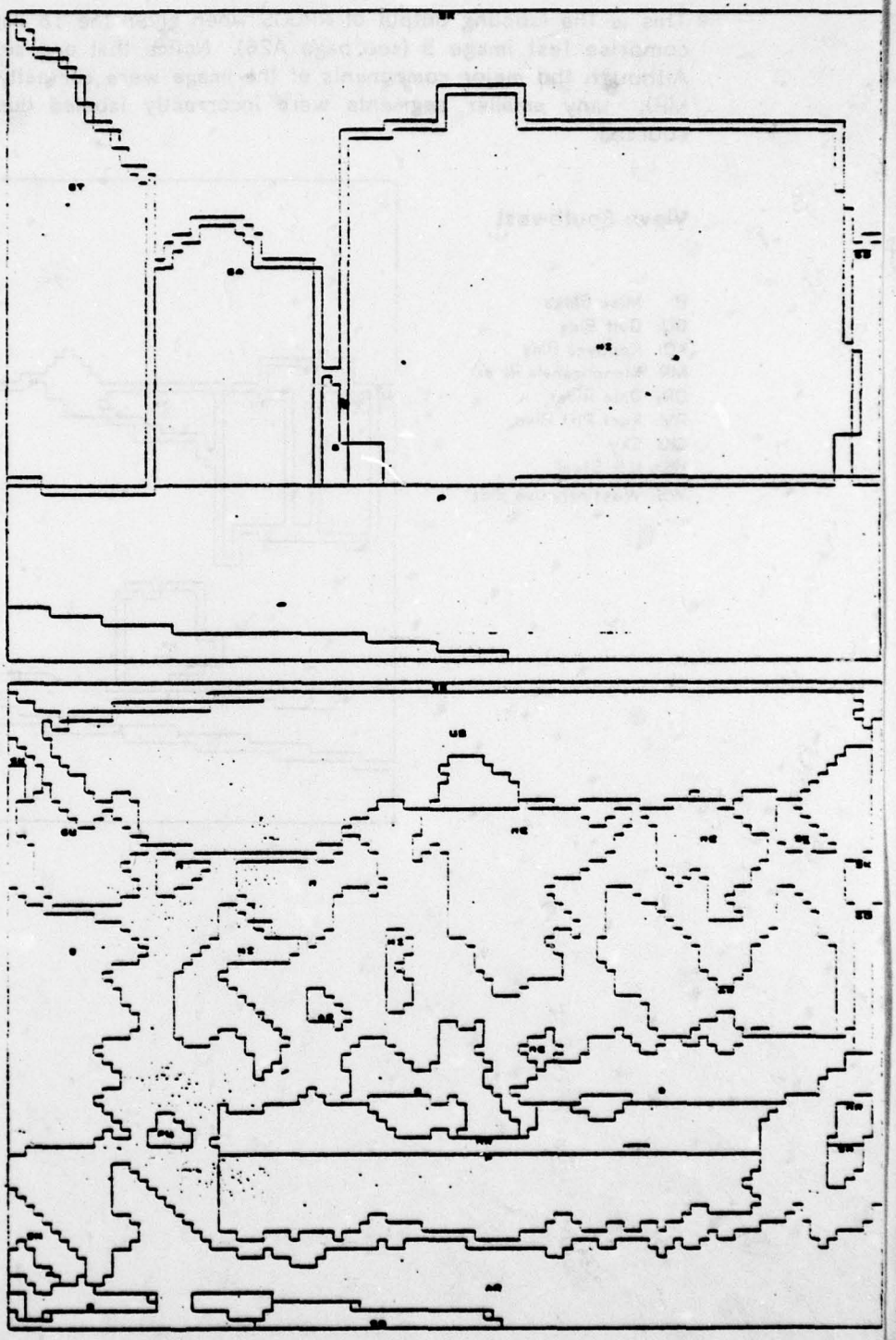
AR: Alleghany River
B: Misc Bldgs
DB: Duquesne Bridge
EQ: Equibank
GA: Gateway 1
GB: Gateway 2
GC: Gateway 3
GT: Gateway Towers
HI: Hilton Hotel
M: Mountains
MR: Monongahela River
OO: One Oliver Plaza
P: Park
PB: Fort Pitt Bridge
PN: Pittsburgh National Bank
PR: Pittsburgh Press
PV: Fort Pitt Blvd
R: Misc Roads
SK: Sky
SN: Snow
SO: State Office Bldg
ST: Stanwix Bridge
WP: Westinghouse Plaza
WS: Westinghouse Bldg



APPENDIX VI: Human and ARGOS Labeling, Test Scene 8

View: Point

AR: Allegheny River
B: Misc Bldgs
FU: Fulton Bldg
GA: Gateway 1
GD: Gateway 4
GT: Gateway Towers
GU: Gulf Bldg
HI: Hilton Hotel
M: Mountains
ME: Mellon Bank
MR: Monongahela River
P: Park
SB: 6th Ave Bridge
SK: Sky
SN: Snow
US: U.S. Steel

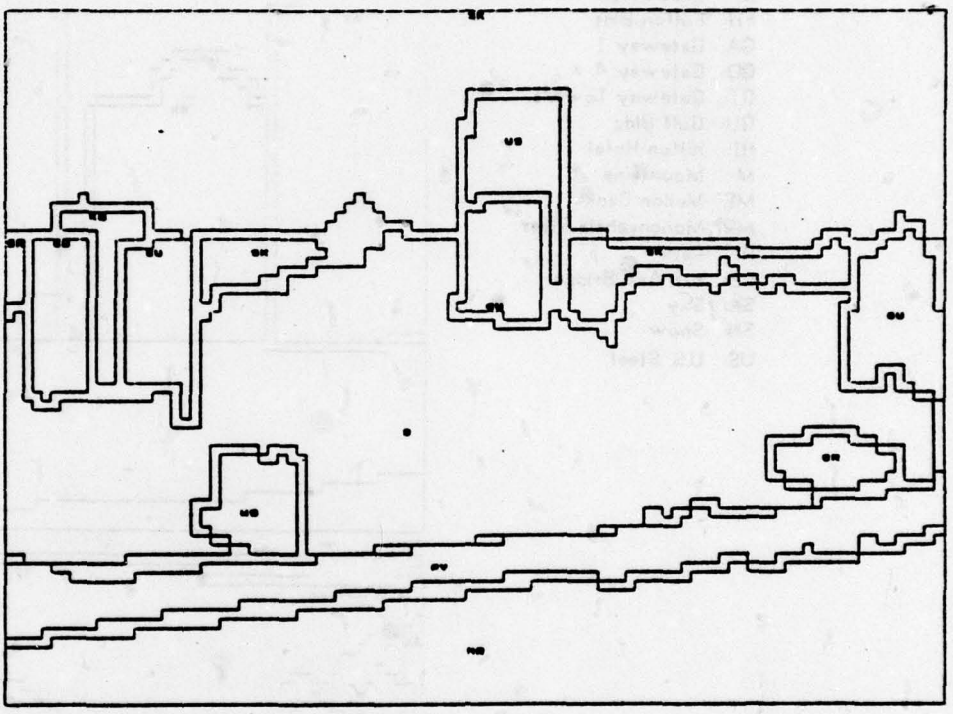


APPENDIX VII: ARGOS Labeling of Hand Segmented Test Image 5

This is the labeling output of ARGOS when given the 16 hand-drawn segments which comprise test image 5 (see page A26). Notice that one segment was left unlabeled. Although the major components of the image were correctly identified (SK, US, B, PV, MR), many smaller segments were incorrectly labeled due to incorrect knowledge sources.

View: Southwest

B: Misc Bldgs
GU: Gulf Bldg
KO: Koppers Bldg
MR: Monongahela River
OR: Ohio River
PV: Fort Pitt Blvd
SK: Sky
US: U.S. Steel
WS: Westinghouse Bldg

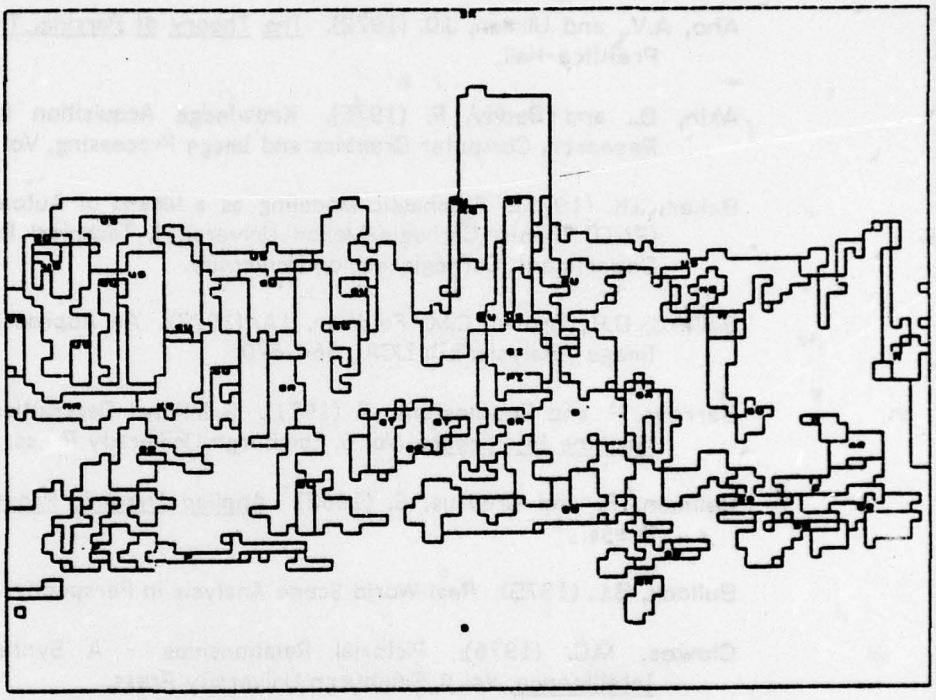


APPENDIX VIII: ARGOS Labeling of Machine Segmented Test Image 5

This is the labeling output of ARGOS when given the 67 segments produced by an automatic clustering algorithm (Shafer and Kanade, 1978). ARGOS was not trained to handle these automatic segmentations, so the labeling quality is poor. This result was generated as the thesis went to press and is included only for the sake of completeness.

View: Southwest

AR: Allegheny River
B: Misc Bldgs
DB: Duquesne Bridge
GA: Gateway 1
GC: Gateway 3
GI: Gimbles
GR: Grant Bldg
GT: Gateway Towers
GU: Gulf Bldg
KO: Koppers Bldg
M: Mountains
ME: Mellon Bank
MR: Monongahela River
OL: Oliver Bldg
OR: Ohio River
P: Park
PI: Pick Roosevelt Hotel
SK: Sky
SN: Snow
US: U.S. Steel



REFERENCES

Aho, A.V., and Ullman, J.D. (1972). The Theory of Parsing, Translation, and Compiling, Prentice-Hall.

Akin, O., and Reddy, R. (1976). Knowledge Acquisition for Image Understanding Research, Computer Graphics and Image Processing, Vol 6, 307-335 (1977).

Baker, J.K. (1975). Stochastic Modeling as a Means of Automatic Speech Recognition, (Ph.D. Thesis, Carnegie-Mellon University), Technical Report, Computer Science Department, Carnegie-Mellon University.

Ballard, D.H., Brown, C.M., Feldman, J.A. (1977). An Approach to Knowledge-Directed Image Analysis, 5th IJCAI, 664-670.

Barrow, H. and Popplestone, R. (1971). Relational Descriptions in Picture Processing, Machine Intelligence, Vol 6, Edinburgh University Press.

Bellman, R. and Dreyfus, S. (1962). Applied Dynamic Programming, Princeton Univ. Press.

Bullock, B.L. (1975). Real World Scene Analysis in Perspective, ACM-75, 25-28.

Clowes, M.C. (1976). Pictorial Relationships - A Syntactic Approach, Machine Intelligence, Vol 4, Edinburgh University Press.

Ejiri, M., Uno, T., Yoda, H., Goto, T., and Takeyasu, K. (1971). An Intelligent Robot with Cognition and Decision-Making Ability, 2nd IJCAI, 350-358.

Erman, L. and Lesser, V. (1975). A Multi-Level Organization for Problem-Solving Using Many Diverse, Cooperating Sources of Knowledge, 4th IJCAI, 483-490.

Feldman, J.A., and Yakimovsky, Y. (1975). Decision Theory and Artificial Intelligence: I. A semantics-Based Region Analyzer, Artificial Intelligence, Vol 5, 349-371.

Fischer, C., Drawings and character design.

Fischler, M.A., and Elschlager, R.A. (1973). The Representation and Matching of Pictorial Structures, IEEE Transactions on Computers, C-22, #1, 67-92.

Forney, G.D. (1973). The Viterbi Algorithm, Proceedings of the IEEE, Vol 61, #3, 268-278.

Fu, K.S. (1976). Syntactic Pattern Recognition, Digital Pattern Recognition, (Fu, ed.), Springer Verlag.

Goodman, R.G., Personal communications.

Harlow, C.A. (1973). Image Analysis and Graphs, Computer Graphics and Image Processing, Vol 2, 60-82.

Homer (700 B.C.). The Odyssey.

Hu, M.K. (1961). Pattern Recognition by Moment Invariants, Proc. IEEE, Vol 49, 1428.

Kanade, T., Personal communications.

Kender, J., Personal communications.

Keng, J. and Fu, K.S. (1976). A Syntax-Directed Method for Land-Use Classification of LANDSAT Images, Symposium on Current Math. Problems in Image Science, Nov 10-12, Monterey, CA.

Kettig, R.L. and Landgrebe, D.A. (1976). Classification of Multispectral Image Data by Extraction and Classification of Homogeneous Objects, IEEE trans on Geoscience Electronics, Vol 14, 1, 19-26.

Levine, M.D. (1977). A Knowledge-Based Computer Vision System, to appear in Riseman and Hanson, forthcoming book on vision, Academic Press.

Lowerre, B.T. (1976). The HARPY Speech Recognition System, (Ph.D. Thesis, Carnegie-Mellon University), Technical Report, Computer Science Department, Carnegie-Mellon University.

Lowerre, B.T. and Reddy, D.R. (1977). Representation and Search in the Harpy Connected Speech Recognition System, C-MU Movie.

Mackworth, A.K. (1977). On Reading Sketchmaps, 5th IJCAI, 598-606.

Minsky, M. (1975). A Framework for Representing Knowledge, The Psychology of Computer Vision, (P. Winston, ed.), MIT Press, 211-277.

Narasimhan, R. (1966). Syntax-Directed Interpretation of Classes of Pictures, Communications of the ACM, Vol 9, 3, 166-173.

Nevatia, R. and Binford, T.O. (1977). Description and Recognition of Curved Objects, Artificial Intelligence, Vol 8, 77-98.

Nilsson, N. (1971). Problem Solving Methods in Artificial Intelligence, McGraw Hill.

Ohlander, R.B. (1975). Analysis of Natural Scenes, (Ph.D. Thesis, Carnegie-Mellon University), Technical Report, Computer Science Department, Carnegie-Mellon University.

Perkins, W.A. (1977). Model-Based Vision System for Scenes Containing Multiple Parts, 5th IJCAI, 678-684.

Price, K. (1976). Change Detection and Analysis of Multispectral Images, (Ph.D. Thesis, Carnegie-Mellon University), Technical Report, Computer Science Department, Carnegie-Mellon University.

Reddy, D.R. and Rubin, S. (1978). Representation of Three Dimensional Objects, Technical Report, Computer Science Department, Carnegie-Mellon University.

Rodd, E.M. (1972). Closed Boundary Field Selection in Multispectral Digital Images, IBM Pub 320.2420.

Rosenfeld, A., Hummel, R.A., and Zucker, S.W. (1976). Scene Labeling by Relaxation Operations, IEEE Trans. Syst., Man, Cybern., 420-433.

Sakai, T., Kanade, T., Yu-ichi, O. (1976). Model-Based Interpretation of Outdoor Scene, 3rd International Joint Conference on Pattern Recognition, 581-585.

Shafer, S. and Kanade, T., 1978. KIWI: A Flexible System for Region Segmentation, Tech. Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh, Pa. (in preparation).

Tamura, H., Mori, S., Yamawaki, T. (1977). Effectiveness of Textural Features for Classification of Aerial Multispectral Images, Proc. IEEE conf on Pattern Recognition and Image Processing, 289-298.

Tenenbaum, J.M. and Barrow, H.G. (1976). Experiments in Interpretation-Guided Segmentation, Technical Note 123, Stanford Research Institute, Menlo Park, CA.

Tou, J.T. and Gonzalez, R.C. (1974). Pattern Recognition Principles, Addison-Wesley.

Waltz, D. (1975). Drawings of Scenes with Shadows, The Psychology of Computer Vision, (P. Winston, ed.), MIT Press, 19-91.

Williams, T., Lowrance, J., Hanson, A., Riseman, E. (1977). Model-Building in the VISIONS System, 5th IJCAI, 644-645.

Woods, W.A. (1970). Transition Networks for Natural Language Analysis, Comm. ACM, Vol 13, 591-606.