USC–SIPI REPORT #137

Intelligent Interpretation of Aerial Images

by

V. Venkateswar and R. Chellappa

March 1989

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 400
Los Angeles, CA 90089-2564 U.S.A.

Abstract

The task of interpreting aerial images can be divided into two sub-tasks - low-level and high-level. The purpose of the low-level module is to abstract some primitives (eg: lines, regions) from the raw image data to be input to the high-level module. The high-level module then attempts to synthesize these input primitives into objects using various sources of knowledge. However, the output from the lower level module is frequently ambiguous, fragmented and insufficient. Consequently, there is a search space involved as in many other artificial intelligence problems. Purely algorithmic procedures, that make ad hoc choices when faced with alternatives, are bound to have limited practical use. Some kind of intelligent search mechanism needs to be employed. At the Signal and Image Processing Institute, USC, we are implementing a system for intelligent interpretation of aerial images. The lower-level module is a new line detector. An ATMS-type multiple context truth maintenance system (MCTMS) oversees search in multiple contexts. This search scheme is well suited to aerial image interpretation. Alternative paths can be simultaneously explored and competing contexts can be directly compared. The process of object synthesis is carried out in a hierarchical manner. An object is composed of faces, faces are formed by edge rings, edge rings are formed by edges, which are defined by their end vertices. Each element in this hierarchy is represented by an uniform representational unit - a frame. Relations between these elements are also represented as frames. The frame paradigm permits us to perform expectation driven processing and also facilitates object oriented programming. The domain of interpretation is currently restricted to buildings that are compositions of rectangular parallelopipeds. Currently our system is only partially complete. The system is capable of composing edge rings and can detect closed edge rings.

Contents

List of Figures

1 Introduction

Most vision systems can be conveniently divided into three subsystems; a low level module, an intermediate level module and a high level module. Feedback between the stages is possible (to resolve ambiguities, for instance), but is often not implemented (evident from the summaries of some popular vision systems in Section 5). Since the size of the raw image data is large, the purpose of the low level module is to abstract some primitives (eg: lines, regions) from the raw image data to be input to the higher level modules. The intermediate level module gets a surface based description (a $2\frac{1}{2}$-D sketch) from this input primitives using techniques like stereo matching, shape from shading etc. The higher level module then attempts to infer symbolic descriptions from this sketch. Sometimes the distinction between the intermediate level and the high level module is not so clear, especially if there is a massive amount of complex interaction between this stages. This is the case in our system (described in Section 4). So for purposes of discussion we will assume that there are effectively only two modules in vision systems – low level and high level.

1.1 The Low Level Module

The amount of data contained in a typical aerial image is huge. The purpose of the low level module is to abstract some primitives from this raw image data to be input to the higher level modules. The kind of data to be abstracted depends naturally on the nature of the objects to be detected, the nature of the input images and on the architecture of the higher level modules.

Nagao et al., present a system for the interpretation of high altitude multi-spectral images in [1]. The purpose of the system is to classify a multi-spectral image into different regions (crop fields, roads, buildings etc.). Consequently, the input primitives for the higher level module are *elementary regions* extracted from the image based on spectral properties. In [2], a system for the interpretation of airport images is presented. Typical airport images are characterized by linear features (runways etc.) and regions (like grass regions). An array of image processing tools (a region-growing segmentation program, a road feature follower and a stereo analysis program) are used to abstract data for the high level module. In ACRONYM ([3]), objects are modeled as a restricted class of generalized cones. The low level primitives are ribbons and ellipses. Ribbons in the image are generated by swept surfaces of generalized cones and ellipses are generated by the end surfaces. In [4, 5, 6] the task is to detect buildings that are bound by planar surfaces. Typical images exhibit a high contrast along the edges of the buildings. The input primitives are naturally, linear segments extracted from the image. The system described in Section 4 uses a similar model for buildings. The input primitives for this system are linear segments that are extracted using a simple and efficient line segment extractor described in Section 2.

1.2 The High Level Module

The output from the current low level modules is frequently fragmented, ambiguous and insufficient. This could be attributed to many factors. First, the current, state-of-the-art low level modules are far from perfect. However, this is not entirely the source of the problem. Since an image is the projection of a 3-D scene onto a plane, there is a loss of information. Occlusions can obliterate lines and regions that should be visible in the image. For example, we cannot assume that all the lines in the image are caused by object and shadow edges. Surface markings can also form lines in the image. We cannot find perfect models for some objects because they come in a continuum of shapes. We have to come up with a mathematically tractable model that can describe this class of objects. This is bound to create further uncertainty when synthesizing objects from an image.

Given the nature of the input data, pure algorithmic approaches that make ad hoc choices when faced with ambiguity are bound to fail miserably. An algorithmic approach is very similar to a hill climbing approach where the most promising path is followed and the other paths are forgotten ([7]) [1]. Both approaches suffer from a lack of guarantee that they will be effective. The only advantage they share is that they are less combinatorially explosive than comparable global methods ([7]). Both methods are not advisable in the context of the highly ambiguous and noisy nature of vision data. Other means of combating this combinatorial explosivity need to be employed. Domain knowledge can be used to limit the size of the graph to be searched. In addition to domain knowledge, the high level module described in Section 4, uses a Multiple Context Truth Maintenance System (MCTMS) to conduct an efficient and coherent search of a problem graph. Previously, Provan in [8] reported using the ATMS ([9, 10]) for an artificially constructed task of identifying puppets from sets of rectangles. The research reported here uses an ATMS-type MCTMS for the much more complex task of interpreting real aerial images. In addition, we use frames as our knowledge representation formalism. All programming objects and the relations between them are represented by frames. This object-oriented paradigm facilitates system construction, modification and extension.

In Section 3 we describe some high level techniques used by our system. We also com-

[1]However, an algorithm has no concept of recorded states (or intermediate steps). Intermediate steps can be explicitly recorded in a hill climbing approach. This could be of aid in debugging the system code. This could also allow backtracking to some previous state, generating all possible paths and following a path that is not the same as the path already examined.

pare and contrast symbolic approaches to connectionist (and other numerical approaches). A system that is being built using these high level techniques is described in Section 4. In Section 5 we present of an overview of some related vision systems. We finally conclude in Section 6.

2 Line Extraction

Several methods are known in the literature ([11, 12] etc.) for the extraction of straight line segments from images. In the Nevatia-Babu method ([11]), edge pixels (after thinning and thresholding) are linked by marking the locations of the predecessor and the successor of each edge element in a predecessor and a successor file respectively. The boundary segments are then computed from these files. Each boundary segment is then approximated by a series of piecewise linear segments using a variation of the well-known iterative end point fitting method.

In Burns et al. [12], edge pixels are first determined by convolution with two simple 2×2 masks. The pixels are then grouped into line–support regions of similar gradient orientation. Then the intensity surface associated with each line–support region is approximated by a planar surface. Straight lines are then extracted by intersecting this fitted plane with a horizontal plane representing the average intensity of the region weighted by a local gradient magnitude.

The Nevatia-Babu Line Finder and the Burns Line extractor use simple edge detection schemes as front ends for their line extractors. Consequently, the process of linear feature extraction is quite involved, especially in the latter. Recently there has been considerable progress in edge detection. In [13], we have compared several of the newer edge detection schemes. A simple and natural straight line extractor we describe below is sufficient to extract the lines in the image if a high quality edge detector is used. We used the edges produced by the Canny edge detector ([14]) as input to our line extractor. In our method, each edge pixel is given a line label based on a continuity criterion. This will lead to generation of a label image. Corresponding to each pixel on the edge image there is a line label on the label image. Edge pixels that lie on the same line will get the same line label. Also associated with each line label are certain macro descriptors of the line. The label image and the line descriptors can be used to further link fragmented lines and in postprocessing steps to remove noisy lines.

2.1 A simple line extractor

In our system, an edge image is first generated using the Canny edge detector ([14]). The direction of each edge pixel is quantized into one of eight directions, as shown in Figure 1. During linking the image is scanned left to right and top to bottom (LRTB). Each edge pixel is assigned a line label when it is visited. For this, preset templates based on edge directions (Figure 1) test if the current edge pixel is a continuation of any line in the immediate neighborhood. The templates for three of the eight directions are shown in Figure 1. The squares in the figure correspond to pixels. The dark square corresponds to the pixel currently being visited. Let us call the pixel in the lighter square as the *label pixel.* As can be seen from the figure, the templates check if the current pixel is a continuation of the line corresponding to the label pixel. The templates are tested for, in the order presented. When a template matches, all the participating pixels are given the line label of the label pixel. If no templates match, it can be assumed that this pixel belongs to a new line, so it is assigned an unused, new line label. Figure 2 further illustrates this scan-and-label process on a simple sample picture.

At the end of the LRTB scan, each edge pixel has a line label associated with it. Edge pixels that belong to the same linear segment will have the same line label. Corresponding to the edge image, there is now a label image. This label image will prove useful in the later stages of the linking process, mainly as a spatial index for the lines.

The process of line extraction can be viewed as a serial connected components algorithm. The edge pixels are the nodes in the graph. Nodes that fit the templates shown in Figure 1 are connected. At the end of the algorithm, the graph is split into a number of lines. A label assigned to each node identifies the line (connected component) it belongs to.

We also associate a record with each line label. This record has fields for the end point coordinates of the line, a field for the average edge contrast along a line, a field for pixel support of the line, a field for the line direction and one for the line length. The last two fields are redundant. But they serve to cache values that are repeatedly used in subsequent processing. When an edge pixel is assigned a line label, the fields of the record corresponding to that line label are appropriately updated.

It should be clear from the above description and the templates in Figure 1 that the decisions to link are made locally and conservatively. Consequently, there are virtually no wrongly linked lines in the generated description. However, weak lines are frequently fragmented. So we link significantly colinear line fragments, based on Lowe's ([15]) colinearity significance measure. To link colinear lines, we first examine a small neighborhood at the end points of each line on the label image. We then compute Lowe's colinearity measure for lines with a similar direction in this neighborhood. If the significance is greater than a certain threshold then the two lines are merged. The merging process is continued recursively till no more merging is possible. Lines with only a single pixel support are deleted.

A typical aerial image is shown in Figure 3. The edges detected in the image are shown in Figure 4. The Canny edge detector ([14]) was used. The edge image is given as the input to the linear feature extractor described in this section. After the first stage of linking the lines shown in Figure 5 are obtained. The number of lines in this figure is 933. After deleting lines with a single pixel support and linking significantly colinear linear segments based on Lowe's colinearity measure the 479 lines in Figure 6 are obtained The arrow heads in Figure 5 illustrate how the lines separate regions of low and high contrasts. The region of higher contrast is to the right of the arrow. No arrow heads are shown in Figure 6 so that it can be easily compared to Figure 7, which shows the lines detected by the Nevatia-Babu line finder ([11]).

Quite a few lines in Figure 6 do not correspond to either object edges or their corresponding shadow edges. Most of such lines can be eliminated by deleting short lines that are isolated, i.e. do not have any significantly long lines in their vicinity. For this, we examine a small neighborhood around the short lines on the label image. The lines can also be thresholded based on functions of their length and contrast. The purpose of these line reduction steps is only to reduce the size of the search space that has to be explored by the higher level module. Theoretically these cleaning steps are unnecessary. But computationally it makes little sense to work with a huge database of lines, where a considerable number of the lines are of no value to the problem solving process.

After deleting small and isolated lines and thresholding the lines based on their length and average contrast, the lines shown in Figure 8 are obtained. The number of lines is reduced to 105 in this step. Effectively these postprocessing steps reduced the number of lines from 479 to 105; a significant reduction. This reduction could lead to considerable memory and speed savings in the high level module.

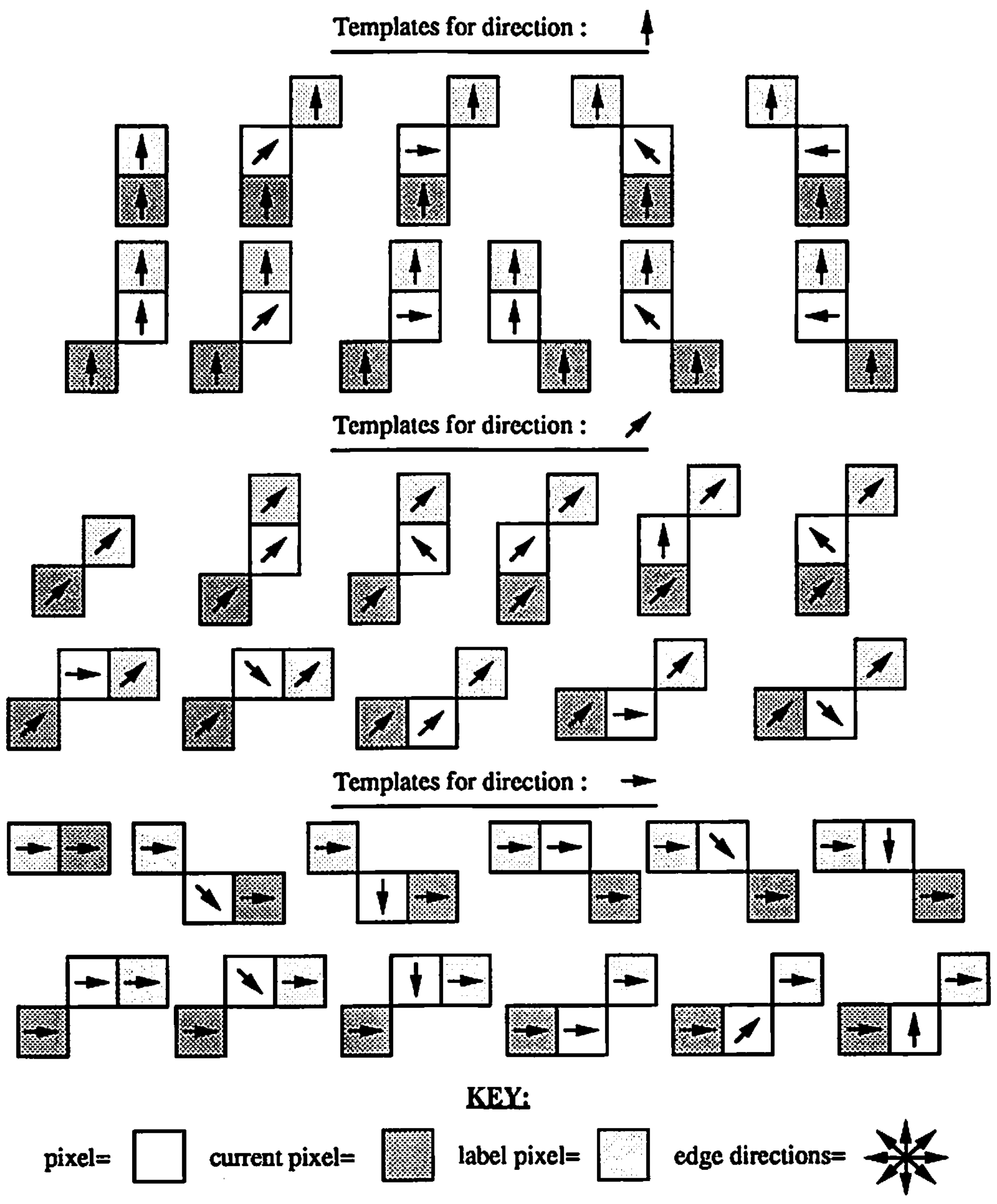
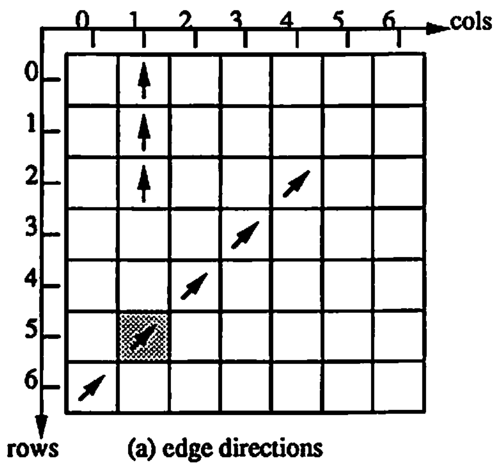

Templates for direction :
Templates for direction :
Templates for direction :
KEY:
pixel=
current pixel=
label pixel=
edge directions=

Figure 1: Templates for linking.



(a) edge directions

0	3	0	0	0	0	0
0	4	0	0	0	0	0
0	3	0	0	5	0	0
0	0	0	5	0	0	0
0	0	5	0	0	0	0
0	9	0	0	0	0	0
8	0	0	0	0	0	0

(b) edge magnitudes

0	1	0	0	0	0	0
0	1	0	0	0	0	0
0	1	0	0	2	0	0
0	0	0	2	0	0	0
0	0	2	0	0	0	0
0	▨					

line-1 =
{start = (0, 1); end = (2, 1);
pixel-support = 3; avg-contrast= 3}

line-2 =
{start = (2, 4); end = (4, 2);
pixel-support = 3; avg-contrast= 5}

(c) the label image and line records just before visiting the dotted pixel

1	0	0	0	0	0	0
1	0	0	0	0	0	0
1	0	0	0	2	0	0
0	0	0	2	0	0	0
0	0	2	0	0	0	0
0	2					

line-1 =
{start = (0, 1); end = (2, 1);
pixel-support = 3; avg-contrast= 3}

line-2 =
{start = (2, 4); end = (5, 1);
pixel-support = 3; avg-contrast= 6}

(d) the label image and line records just after visiting the dotted pixel

Figure 2: The scan and label process.

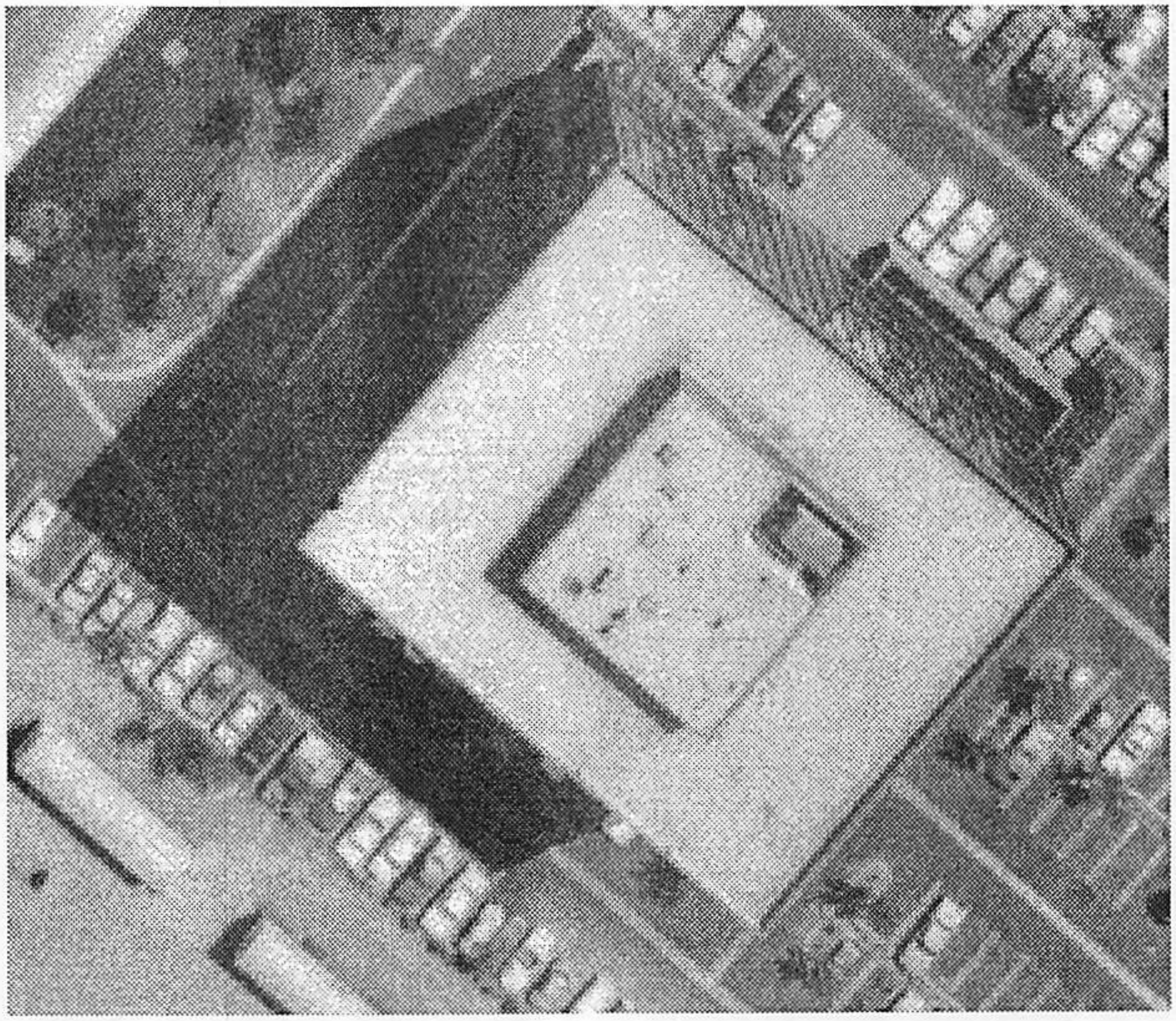


Figure 3: A typical aerial image.

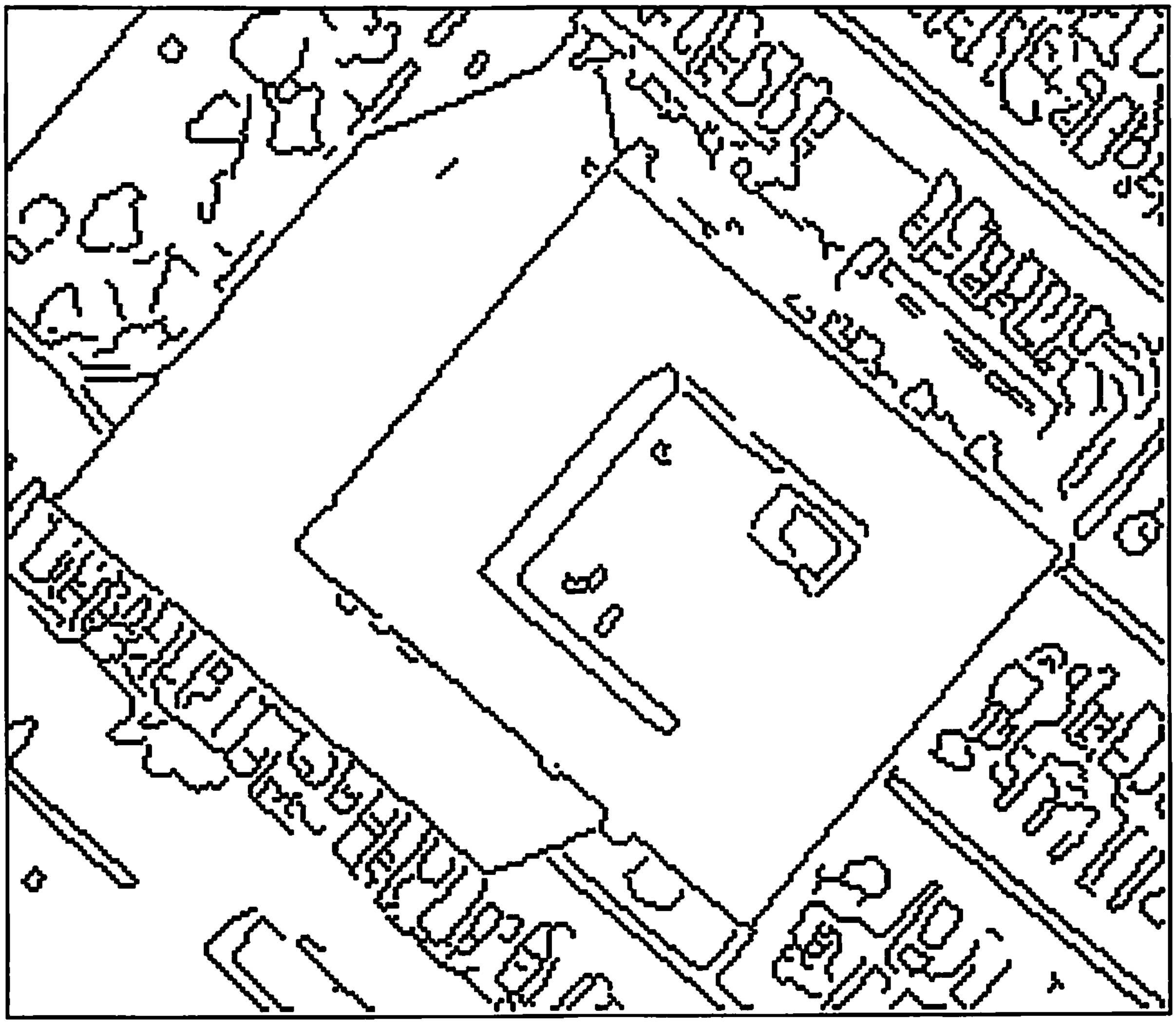


Figure 4: Edges detected in the aerial image.

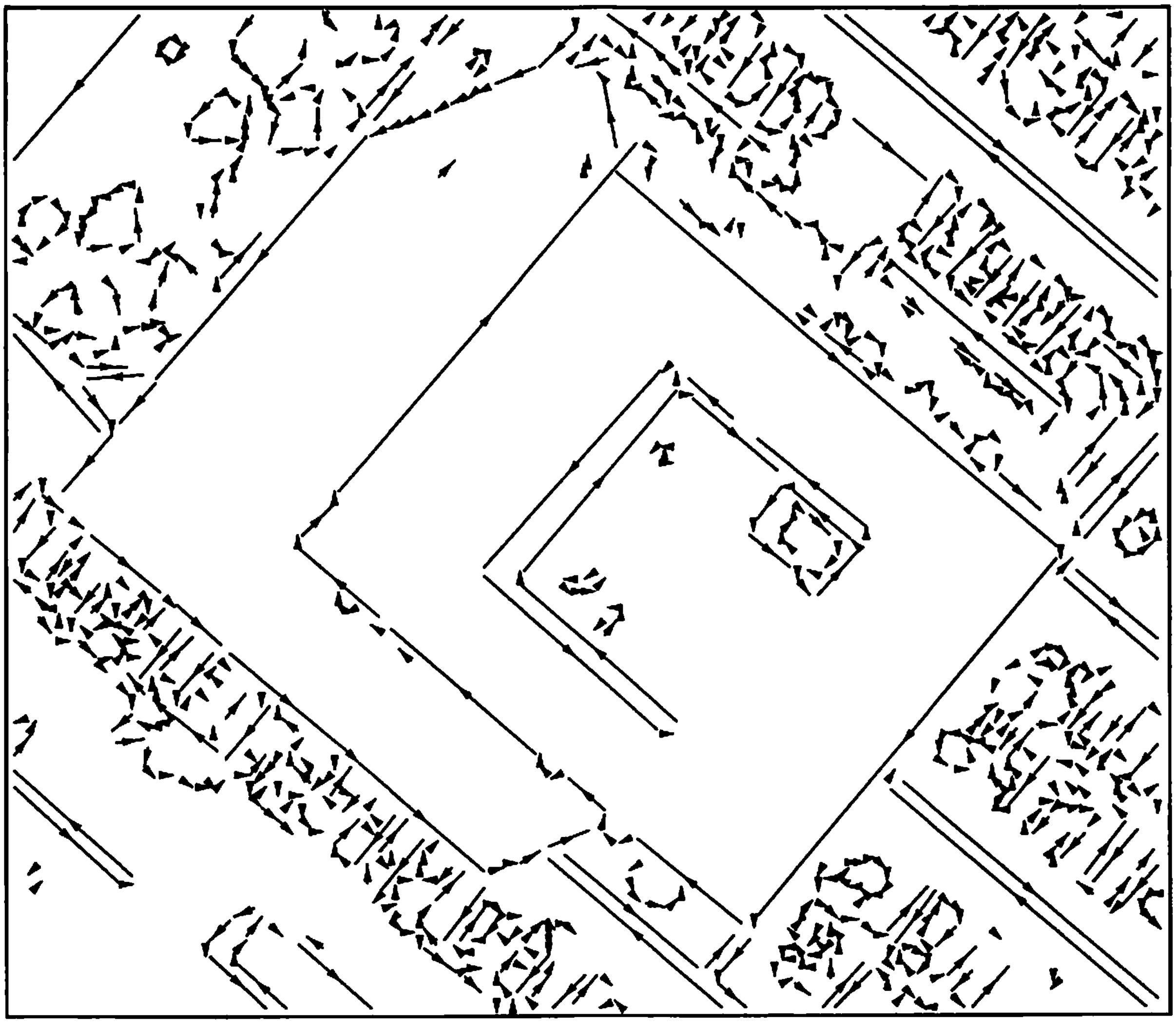


Figure 5: Set of lines before linking significantly colinear line segments.

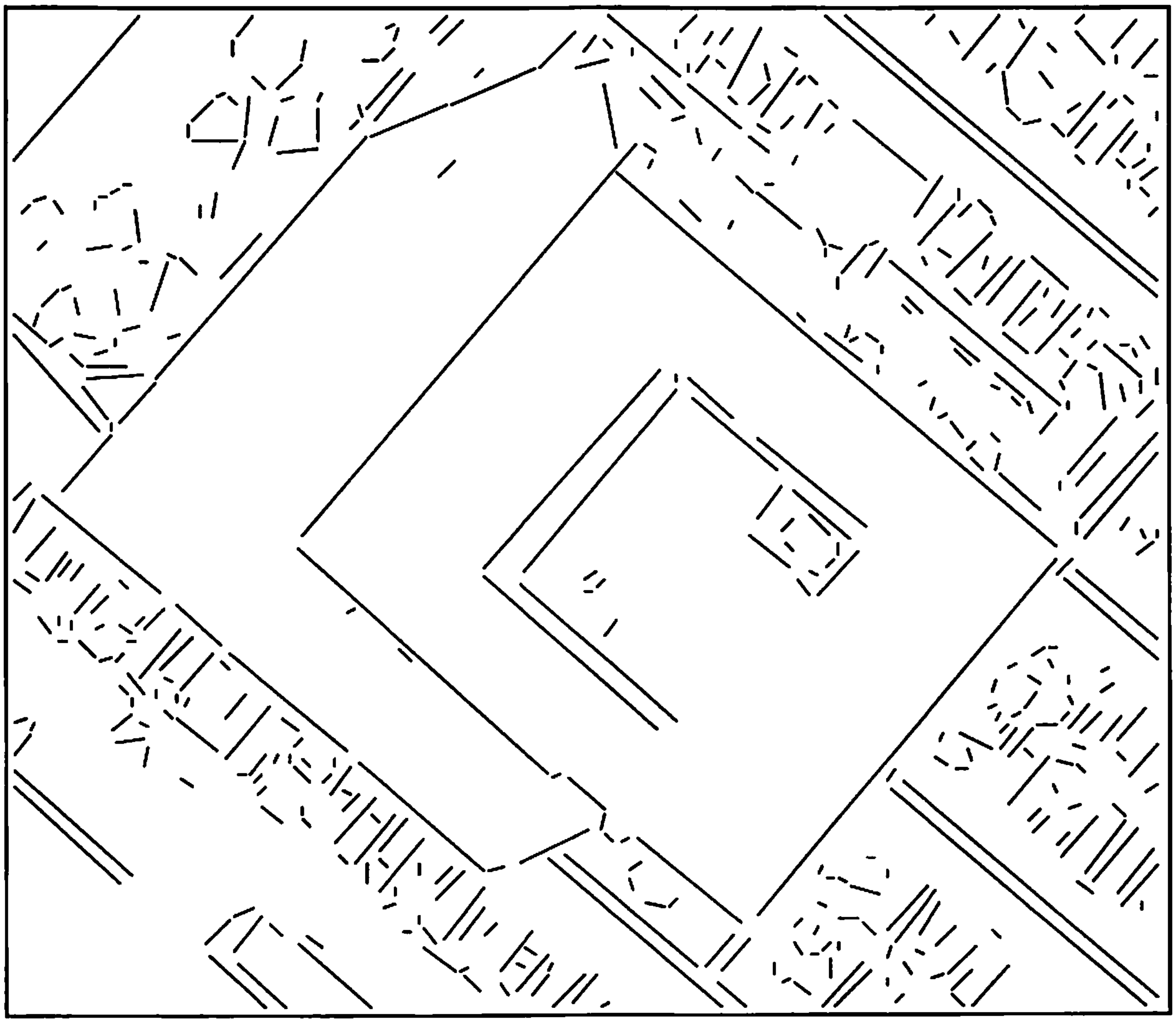


Figure 6: Set of lines after linking significantly colinear line segments.

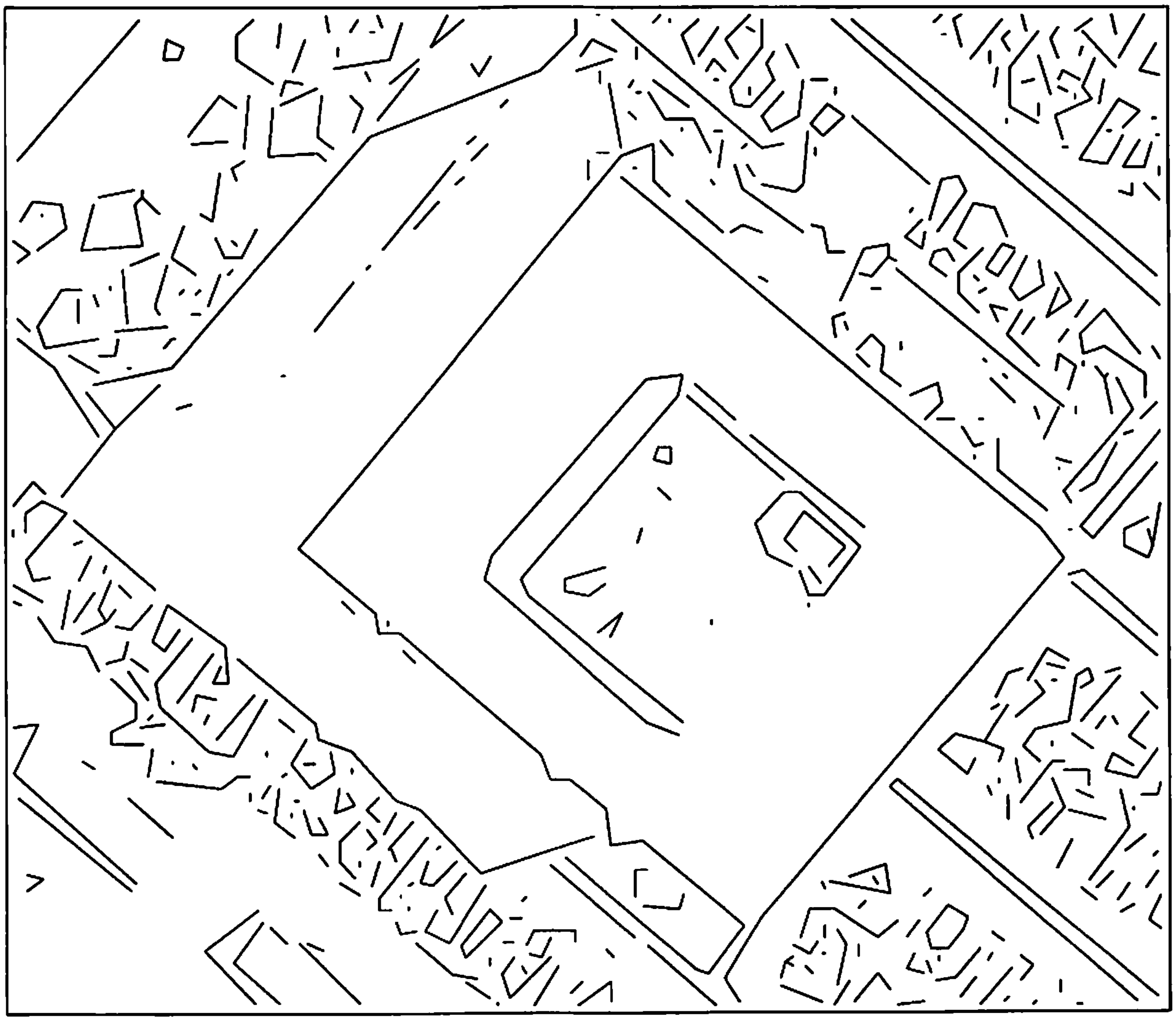


Figure 7: Set of lines obtained by the Nevatia-Babu Line Finder.

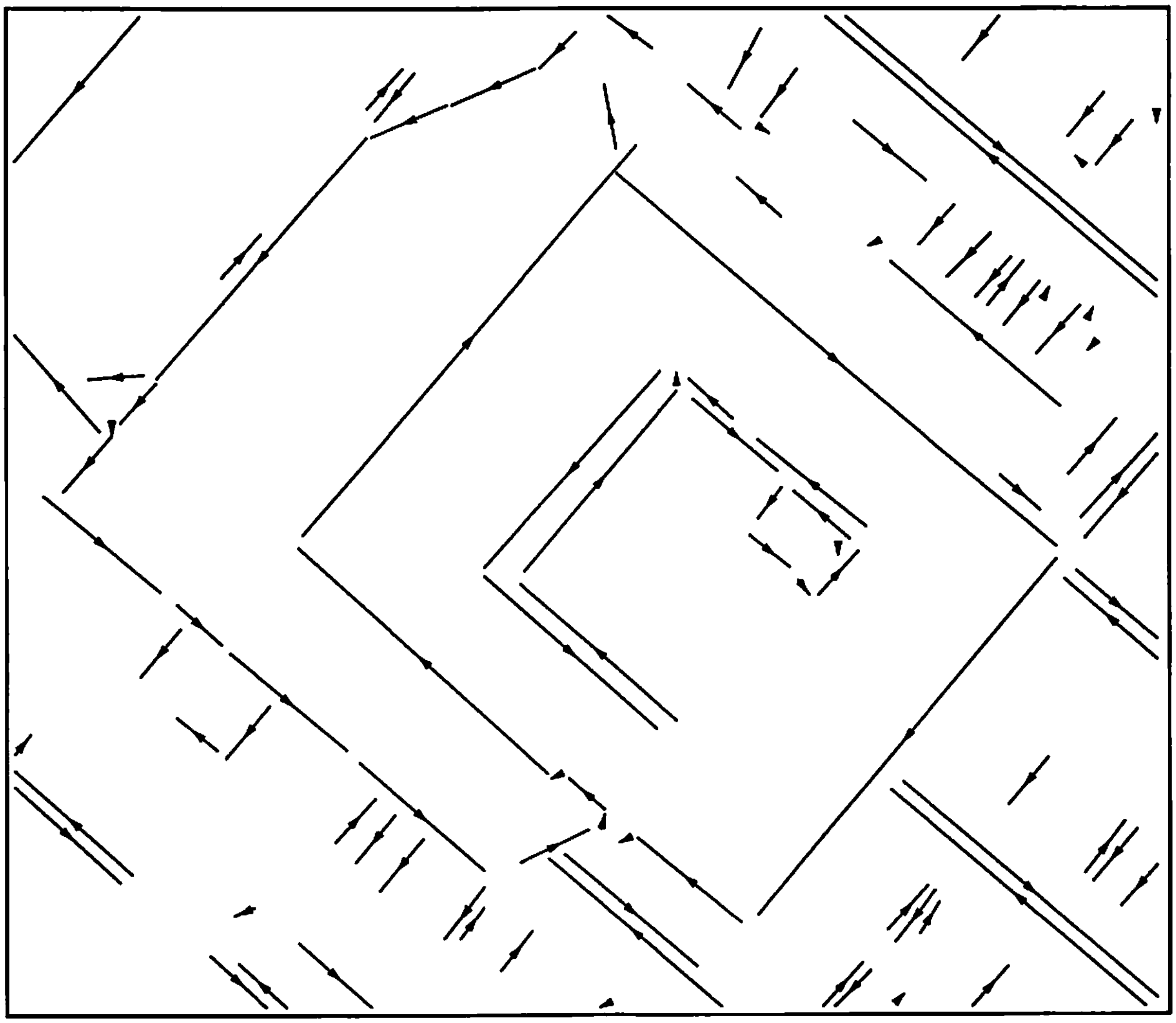


Figure 8: Final set of lines.

3 Overview of the high level techniques

In this section we describe some of the high level techniques we employ for the system described in Section 4. We also try to enunciate the motivation behind these approaches.

3.1 Search using Multiple Context Truth Maintenance

In vision, as in many other domains, we frequently encounter situations where there are alternate paths in the problem solving process. Based on current evidence, all these paths may appear to be equally likely. Essentially, there is a search space involved. It is the task of the problem solver to find the correct solution in this search space, intelligently and efficiently. Many existing vision systems are however mostly algorithmic. The fundamental issue of how to deal with alternatives is not addressed in a satisfactory manner. Section 5 describes some current vision systems and how they succeeded or failed in satisfactorily addressing this issue. Frequently, one path is preferred over the others based on the local situation. There is no search. There is no guarantee that the correct solution will be obtained.

Consider the simple example in Figure 9. The task is to compose rectangles from the lines in the figure. Some of the lines are fragmented. One way to compose rectangles is to first identify potential corners and then group them. Just based on the local situation near

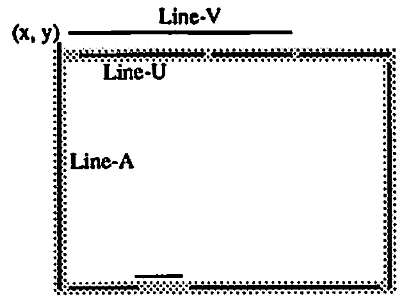


Figure 9: Composing rectangles from lines.

coordinate (x,y), it may look as if the corner formed by Line–A and Line–V is preferable to the one formed by Line–A and Line-U. Line–V is much longer than Line–U and closer to an end point of Line–A. An algorithmic program based on such a heuristic will fail.

Situations more complex than that in Figure 9 are frequently encountered in vision (in some cases it may not even be possible to come up with reasonable heuristics). The reasons are manifold. One reason is because of accidental juxtaposition of events. In Figure 9, Line–V may not be in any way related to any of the other lines in the figure, though it is proximate to some of the other lines and is very similar. Another source of trouble is that the input primitives produced by the low-level modules are frequently noisy and ambiguous. This could be both due to the inadequacy of the current low-level techniques and other imaging issues like occlusion, quality of the camera etc. Another source of problem is in the modeling process itself (for eg: are step edges adequate enough to describe all possible object discontinuities of the image? Most of the edge detectors are step edge detectors). Some completeness is sacrificed by the image modeling assumptions. This adds to the problems faced by the high level module.

The motivation for pure algorithmic approaches is speed and memory savings. The search space in most vision tasks is potentially combinatorially explosive. Depth first search could take too much time, breadth first search too much memory. Vision researchers have even avoided doing some kind of best-first search. For example in the $3-D$ MOSAIC ([4]), a beam search [2] is used to find matching junctions in a stereoscopic pair of images. Another strategy is to let the system infer all the conflicting interpretations in a single context and worry about resolving the conflicts later. SPAM ([2]) uses special conflict resolution rules to sift through conflicting interpretations.

The whole issue of exhaustive symbolic search can be side-stepped by mapping the problem into a form solvable by techniques like relaxation labeling ([16]), constraint satisfaction networks ([5]) etc. ACRONYM ([3]) uses a relaxation algorithm to match maximal subgraphs of the *observation graph* to subgraphs of the *interpretation graph.* CANC ([5]) uses relaxation on a constraint satisfaction network to derive a unique hierarchical network from a network of mutually competing and supporting *collations.*

In relaxation labeling ([17]), each node is assigned all possible labels. Nodes are connected locally. Locally inconsistent labelings are deleted iteratively till (theoretically) there is a single consistent solution. There is no notion of explicit global semantics ([8]). This technique is extremely useful where all possible labelings are known before hand and the data is correct (eg: Waltz's [18] relaxation labeling for line drawings).

When using constraint satisfaction networks ([19]), the objective is to minimize some global algebraic measure. Convergence is not guaranteed with arbitrary global measures and with arbitrary methods of local updating. The problem is converted from a symbolic problem to a numerical problem. The system designer has to come up with an accept-

[2]Beam search is a search technique that is based on breadth-first strategy. Only a limited number of paths are explored. Path pruning is done based on the information available till that point.

able set of numeric weights. Any intelligence is effectively "compiled" into these numeric weights. So reasoning cannot be done with the knowledge stored.

However, in a lot of cases it may not be possible to come up with a natural mapping scheme that preserves all domain knowledge and allows intelligent and explicit interpretation. With these techniques, there is no notion of explicit and intelligent reasoning. There are no simple ways to compare two possible solutions (for eg: at two different local minima).

We use a two pronged approach to deal with the problem of combinatorial explosion. First, domain knowledge is extensively used to prune the problem search space. Second, an efficient, coherent and opportunistic search is performed in multiple contexts using an ATMS-type Multiple Context Truth Maintenance System (MCTMS). The arguments in favor of an MCTMS search mediation below, are derived from [9, 10, 20, 21].

The simplest search strategy is to follow each path and try all possibilities till a solution is found. This is also the most inefficient. Chronological backtracking can improve the efficiency. An improvement over chronological backtracking is dependency–directed backtracking, first introduced in [21] and later used by Doyle ([20]) in his justification based TMS. For details of the advantages, the reader is referred to [9]. To summarize, chronological backtracking has the problem of rediscovering inferences and contradictions. However, if we maintain dependency records for each inference they can serve as a cache so that there are no rediscoveries. When contradictions are encountered, dependency records can be consulted to backtrack to the most recent selection which actually contributed to the contradiction. This serves as a basis for a TMS.

However, as de Kleer ([9]) explains, conventional single context TMSs have a variety of problems. Mainly, only one context can be considered at a time. This creates problems if the problem solver attempts to do a differential diagnosis, the ability to directly compare and contrast solutions in different contexts. de Kleer proposes a multiple context TMS called the ATMS. In the ATMS all the contexts are simultaneously explored. There is no single-state restriction. This permits differential diagnosis. There is no overzealous contradiction avoidance, switching states is easy and the machinery is less cumbersome ([9]).

In the ultimate analysis all exhaustive search strategies (depth-first, breadth-first, best-first, ATMS-type) are equivalent. An MCTMS will find all the solutions that a pure enumeration technique would. An MCTMS improves the efficiency of the problem solver without sacrificing coherency and exhaustivity.

An MCTMS works by manipulating assumption sets ([9]). Assumptions are primitive data from which all other data is derived. They can be manipulated far more conveniently than the datum sets they represent. Identifying the underlying assumptions of a problem is not a trivial task. In Section 4, we try to carefully identify the reasons for the choice of our basic assumptions.

Provan ([8]) uses the ATMS to solve the artificial problem of detecting puppets in a set of rectangles. We demonstrate the use of a MCTMS for a much more complex task.

3.2 Frame based approach

Frames are data-structures with *slots* that can be filled with *fillers* (very similar to those in other languages like KRL ([22]) and KEE ([23])). Frames can be arranged in inheritance hierarchies. This permits description of prototypical situations or generic objects. Specific objects inherit the slot values of the generic type by default. This has the effect of simplifying the task of the system designer by facilitating object-oriented programming ([24]).

Our use of the term frames shares some similarity to Minsky's ([25]) frames and also schemas, as defined by Arbib ([26]) and others. Schemas are units of computational analysis that are used by cognitive scientists as a bridge between overall behavior description and "neuron-like" low-level elements. At a functional level, the interaction of simultaneous computing agents (schemas) model animal behavior. Weymouth in [27], uses a version of schemas that conforms closely to Arbib's criteria ([28]). However, the coherent search paradigm we advocate in Section 3.1 cannot be achieved with such a radical distribution of control. Like Arbib's schemas, our frames can be instantiated and combined to form new frames. Minsky ([25]) proposed frames as descriptions of prototypical situations with instances inheriting the defaults of the prototype. For example, in visual scene analysis, frames can be associated with typical viewpoints. This use of defaults and prototypes is in the same spirit as our use of frames. However, the frame-system scheme proposed by Minsky seems to be more of an attempt to model common-sense thought (cognitive memory models). Much emphasis is given to the idea that though frame systems are less complete than systems based on logic, they are nevertheless better suited to deal with real world complex problems because of their ability to deal with approximations.

A far less benevolent view of the frame movement is taken by Hayes in [29]. He feels that there have been very few positive contributions from this approach, no new reasoning paradigms and no advance in expressive power that logic cannot deal with. According to him, one positive contribution from the frame-movement is an emphasis on the analysis of the prospects of reflexive reasoning (defaults, caricatures etc). He sees a lot of future in the work on non-monotonic logics ([30]).

Many vision systems (for example, [3, 31, 32, 33]) use frames as a knowledge representation formalism as well as a means of achieving control. Some of these systems, take an extreme viewpoint and embed all control as procedures in frames. This permits specialized local control techniques. This could be problematic if a search space needs to be explored, because with this kind of control, coherent and complete search strategies are not possible. In Section 5 salient features of some of these systems are discussed.

In our system described in Section 4, the main control paradigm is a production system. Partial control is delegated to frames, especially if those operations are side-effects to a main operation (for example, the main action could be deleting a frame. As a side effect this will delete all frames that are lower down in the inheritance hierarchy. There is no need to explicitly search for inheriting frames to delete them).

In the frame language that we use, *methods* can be defined over frames. These methods are also inherited across the inheritance hierarchy along with the slot values. For example, the method *compute-direction* can be defined over a frame that represents a generic line as well as over a frame that depicts a generic plane. In case of the line-frame this method would return the vector along the direction of the line and in the case of the plane-frame it would return the surface normal. The use of such techniques may seem artificial and contrived at first glance. However, careful and meaningful use of these techniques could be of invaluable help in building complex systems. These object-oriented techniques can help in abstracting confusing and complex information into programming entities that are easily maintained and manipulated by the system designer. In addition to *methods*, *demons* could be attached to the slots of the frames. Typical demons are *if-needed* and *if-added* demons. *if-needed* demons are triggered when a slot value is requested and *if-added* demons execute side-effects when a value is added to a slot. For example, consider the frame,

```
(def-frame line-frame
   (start)
   (end)
   (direction)
 )
```

as also a more specific instance of this frame,

```
(def-frame line-12
   (instance-of line-frame)
   (start (0 0))
   (end (10 10))
   (direction)
 )
```

We can attach an *if-needed* demon, *compute-direction* to the direction slot of the generic frame. When the direction slot of the frame, line-12, is accessed, this demon is triggered and calculates the direction as $45°$. This saves computation, as the direction slots are calculated only when needed. For example, if there are n instances of the line-frame and only the directions slots of m lines are ever accessed, where m <<n, then it makes little sense to pre-compute all the values and store them before hand.

Similarly, *if-added* demons can be attached to the start and end slots. These could automatically update the direction slot when a new value is added to these slots. Effectively a frame can be made to monitor itself, freeing the system designer from burdening himself with hundreds of inferences that are potential side-effects to some inference action.

3.3 Symbolic paradigms vs. Connectionist paradigms

Two paradigms have emerged in the AI literature for problem solving tasks; symbolic and connectionist paradigms. In the symbolic scheme, information is presented in the form of labelled symbols. The symbols then undergo transformation into other symbols through an interpretive process. In the connectionist paradigm, information is presented more directly (for example; numeric weights on links between nodes) and affects the processing without undergoing an interpretive process. In this section, we attempt to put our method of aerial image interpretation in proper perspective with respect to this controversy; specifically, why we chose the symbolic paradigm over the connectionist (as well as over other non-symbolic techniques).

Much of the arguments in this section are borrowed from [34], a lucid view on the symbolic, connectionist controversy. As explained in [34], when building a complex system (like an aerial image interpretation system), problem decomposition is very important. This reduces system complexity, permits reusability of parts and increases system robustness by permitting accumulation of intermediate evidence. Problem decomposition cannot be avoided in any paradigm. This fact has been recognized by vision researchers, who have liberally used hierarchical decomposition. In the 3-D MOSAIC system ([4]), the problem of synthesizing buildings from input primitives is cast in a hierarchical framework. Points are composed into lines. Lines are composed into planes which are composed to form the outer surfaces of buildings. A similar approach is also used in our system. In [5], where a connectionist scheme is used, line primitives are composed into perceptually significant groupings. These groupings are further composed into more complex collations resulting in a hierarchy. This problem decomposition constitutes the information processing level abstraction described in [34]. Given this compositional framework, either symbolic or connectionist (or some combination thereof) realizations are possible.

Some of the advantages claimed for the connectionist paradigm include massive parallelism, complete distribution of processing and an ability to learn. Connectionist schemes are intrinsically parallel. However, this does not mean that there is no parallelism at all in symbolic schemes. The MCTMS we discussed in Section 3.1 has certain parallel characteristics because all contexts are simultaneously explored. Another argument in favor of connectionism is complete distribution of processing. This advantage is somewhat diluted by the fact that for complex systems problem decomposition is an essential task, and this precludes building a system that is completely distributed. Some form of organization into components is necessary and unavoidable. Yet another argument, frequently cited as the *raison d'être* for connectionist techniques is their ability to learn (from examples). As explained in [34], if learning is to be accomplished in less than geologic time for complex tasks, an appropriately decomposed initial architecture has to be designed. This is a non trivial task and we are not aware of any reasonably complex vision system that incorporates learning into the connectionist scheme that they employ. Without an automatic learning scheme, distributed representations are of limited practical use, as described in [19]. To add a single piece of macroscopic knowledge in a distributed representation, it is necessary to change the interactions between a large number of microscopic units slightly. Manual tuning is virtually impossible for any reasonably sized problem. Learning is still an active area for research ([19]). An example of one vision system that uses a connectionist scheme for an intermediate level task is [5]. No learning scheme is employed. The interconnection weights are manually assigned. More on this system in Section 5.

To summarize, we do not claim that symbolic approaches are superior to connectionist approaches in general for image understanding tasks. But with the particular problem at hand (aerial image interpretation) and the particular compositional strategy used (vertices to edges, edges to planes, planes to objects) symbolic approaches seem to provide certain computational advantages over the connectionist approaches. Part of the reason is that this problem seems to have a large symbolic content (in the way it is cast) that cannot be ignored. Another part of the reason is that the state-of-the-art in connectionist schemes seems to be not yet satisfactory.

We have argued for a symbolic approach to our problem. In the previous sections we have detailed some desirable characteristics that these symbolic algorithms must exhibit. To wit, they need to permit a coherent search strategy and they should utilize a uniform representational language that simplifies the system designers task. The MCTMS was proposed as an attractive aid to perform an efficient, coherent and simultaneous search in multiple contexts. Arguments were also made for using a frame based object-oriented approach for providing uniformity and lucidity. Further advantages can be derived from a frame-based approach because expectation driven processing is possible. However, we do not advocate a radical distribution of control within frames (as in [31, 32]) because this interferes with the first requirement (the need for coherent search). In the next section we describe an aerial image interpretation system that works in a limited domain, but attempts to build on these ideas.

3.4 Choice of the knowledge engineering toolbox

The techniques we described above have been built into a variety of knowledge engineering toolboxes. In [35] and [36], some of the more powerful knowledge engineering tools are compared. We chose *ART*[3] based on speed and price considerations. *ART* is a hybrid, knowledge engineering toolbox that integrates both rule-based and frame-based approaches. A MCTMS is part of this toolbox.

3.5 Propositions, Frames, Rules and Contexts

In this section, we briefly describe some salient features of the integrated, hybrid knowledge engineering tool that we are using on this project - ART. We have changed some of the nomenclature used in [37] so that the terminology is similar to ATMS-terminology ([9]). This discussion will assume that the reader is familiar with de Kleer's ATMS ([9, 10]) and compares de Kleer's ATMS with ART's MCTMS (called *ViewPoints*). This discussion contains some comments and comparisons that are not found in [37] and are partly derived from [9, 10]. The descriptions given here will suffice to understand the remaining sections. For more details, the reader is referred to [37].

proposition A proposition in the data-base is a predicate calculus like expression. The first term in a proposition is a relation name. The relation can be n–ary, where n is a natural number. An example of a proposition is:

```
(end-points line-111 (3 100) (37 67))
```

frames Frames are data structures that have *slots* that are filled with *fillers*. Two kinds of slots are possible - attribute slots and relation slots. Relation slots serve to link frames together to form a frame network. Both inheritance and non–inheritance relations can be defined. *Methods* can be defined over frames. *Active Values* can be attached to attribute slots. These are are arbitrary procedures that are triggered by certain operations on this slot. An example of a frame is:

```
(def-frame line-111
  (instance-of line-frame)
  (start (3 100))
```

[3]*ART* is a trademark of *Inference* corporation

```
  (end (37 61))
  (contrast 56)
  (part-of edge-8)
)
```

Frame slots are actually translated into propositions with binary relations. Proper logical dependencies are maintained as property lists on facts. For example, when a frame is deleted all it's relational and attribute slots are automatically deleted. This also deletes the corresponding inverse relations. When a slot in a frame that is part of a frame inheritance hierarchy is deleted, the inherited values in the frames lower down in the inheritance hierarchy are automatically deleted.

rules Rules are typical production rules (if–then statements). They are similar to the *consumers* in de Kleer's ATMS. The LHS consists of a conjunction of patterns. Propositions in the database can match the LHS patterns. If all patterns of a rule have matching facts, then the rule is triggered. When the rule fires, the action specified on the RHS side is executed. The RHS can have arbitrary procedures. Procedures like *assert*, *retract*, *modify* etc., act on items in the database. For example, the rule shown below identifies lines that share a start point and asserts that fact into the database.

```
(def-rule
  (end-points ?line-X ?start-X ?end-X)
  (end-points ?line-Y ?start-X ?end-Y)
  ⇒
  (assert (same-start-point ?line-X ?line-Y))
)
```

For efficiency purposes the patterns in the LHS of the rules are compiled into a RETE pattern matching network ([38]). This leads to efficient computation of the conflict set. Understanding the dynamics of this network could be crucial for writing an efficient system. However, for pedagogical purposes it can be safely assumed that the conflict set is somehow computed. Priority values attached to the rules determine the priority of the rule when it is placed on the *agenda*. To minimize rule interaction it is recommended that all rules have the same priority number.

contexts Like de Kleer's ATMS, ART's MCTMS works by manipulating assumption sets. The idea is that, since assumption sets are the primitive data from which all other data are derived, they are more conveniently and quickly manipulated. A set of assumptions constitutes an *environment*. The environment together with all the data that can be derived from it constitutes a *context*.

Like the *environment lattice* in de Kleer's ATMS, ART uses a *context* DAG (Directed Acyclic Graph). Each node in the DAG is a context. The node name characterizes the environment and the context. Child nodes have environments that are supersets of parent nodes. The context DAG is explicitly created unlike the environment lattice in the ATMS. The environment lattice is potentially combinatorial. If there are n assumptions, then there are 2^n nodes in the environment lattice, since an environment lattice has all possible environments. ART however creates nodes for only those environments that are germane to the problem solving process. The size of the context DAG can be further controlled by eliminating the descendants of a ***nogood*** environment. If an intermediate solution is found in a context with respect to an ancestor context, then the part of the DAG between the contexts can be collapsed, reducing the size of the DAG. One advantage of the DAG ([9]) representation is that it serves as a cache of the subset and superset relations. When a contradiction is discovered, it is easy to sweep out all the superset environments.To determine whether a new environment is contradictory, one need examine only the supersets of the antecedents which are contradictory ([9]).

Assertions in the database are labelled by their *extent*. The extent is similar to de Kleer's node *labels*. The extent lists the context that a fact is asserted in and the set of contexts it is retracted in. The *extent* together with the knowledge of the DAG structure specifies what contexts the fact is valid in. Child contexts inherit information from parent contexts. The inheritance assumption, the *extents* and the DAG combine to solve the frame problem [4] in an acceptable way.

Contexts ([9]) can be created on the RHS of a rule by using the *hypothesize* function to add new assumptions to the existing assumption set. A hypothesize statement consists the incremental set of assumptions to create a new environment, followed by facts that can be deduced based on the resulting environment. It is the responsibility of the system designer to identify exactly what the underlying assumptions of a problem solving task is. This can be a very difficult task. In the system presented in Section 4, the basic assumptions are that two vertices are connected by an object edge.

The following rule produces the context structure shown in figure 10. In this example, the = macro on the RHS evaluates it's argument. The first *assert* in the hypothesize macro is used to assert the set of assumptions. Any subsequent *asserts*

[4]The ***frame problem*** is not related to frame-structures. In predicate calculus based systems, ***frame axioms*** infer explicitly that a fact will not change across states (frames). Hundreds of these ***frame axioms*** are required to exhaustively catalog all the facts that do not change across states, effectively cluttering the database. This problem is called the ***frame problem***. Fortunately, most knowledge engineering shells that provide state-based reasoning capabilities are not based on pure predicate calculus. In case of ***ART***, there is an implied inheritance in the context (state) hierarchy which takes care of the ***frame problem***.

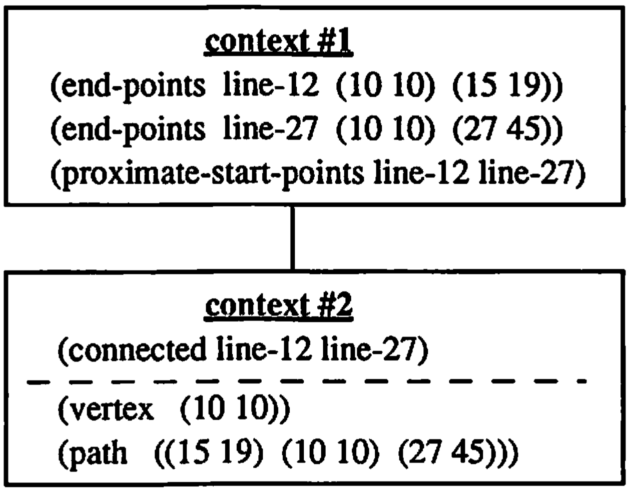

context #1
(end-points line-12 (10 10) (15 19))
(end-points line-27 (10 10) (27 45))
(proximate-start-points line-12 line-27)
context #2
(connected line-12 line-27)
(vertex (10 10))
(path ((15 19) (10 10) (27 45)))

Figure 10: A context hierarchy.

are deductions that are based on these assumptions. The new context created has only one assumption (assuming that it's parent is the root context and therefore has no assumptions) and two deductions based on this assumption. The set with this assumption as it's only element is therefore the characterizing environment of the context. The environment together with all the facts in it's context (including the three propositions inherited from the root) constitutes the complete context.

```
(def-rule
  (end-points ?line-X ?start-X ?end-X)
  (end-points ?line-Y ?start-Y ?end-Y)
  (proximate-start-points ?line-X ?line-Y)
  =>
  (hypothesize
    (assert (connected ?line-X ?line-Y))
    (assert
      (vertex =(* 0.5 (+ ?start-X ?start-Y))
      (path ?end-X =(* 0.5 (+ ?start-X ?start-Y)) ?end-Y))))
)
```

Contexts can be merged creating new contexts (with new environments that are supersets of their ancestors). ART automatically creates merges of contexts to create environments that all the patterns on the LHS side of a rule are valid in. A rule fires in the most general context it is valid in. Resulting actions (like *assert*, *retract* etc.)

are inherited by all it's child contexts. Thus reinferencing in child environments is avoided.

When a contradictory situation is detected in a context then the context node in the DAG is marked as contradictory. All it's descendants are automatically deleted and the context dag pruned. A node that inherits from this node will never be created. On the other hand, if there are reasons to believe that a particular context represents a solution (or an intermediate solution), it can be *confirmed* with respect to an ancestor, collapsing the confirmed context node into it's ancestor's. These two operations of contradicting and confirming are to be wisely used by the problem solver to prevent the combinatorial explosion in the number of contexts.

comparison with the ATMS ART has a lot of similarities with the ATMS ([10]) as can be seen from the previous section which compares some functions. In addition to these differences, the way they deal with multiple justifications and scheduling (control) is different.

Multiple justifications can be problematic in ART. If a problem solver datum is true in 2 nodes that have no elements in common in their base environment then two nodes are created for the same datum.

The ATMS has some methods to exercise some degree of control over the problem solver activity. For example, the problem solver concentrates on nodes with fewer assumptions first. This avoids having to recompute labels when a more specific instance is discovered before a general instance. There is no such preference in ART. In fact, rules on the agenda are activated at random in ART. In the case where a more general context is discovered for a proposition, the proposition node is retracted and a new one is asserted with the changed label. This may retrigger previously fired rules. It is the responsibility of the designer to design the system in such a way that this retrigerring does not affect the reasoning processing but only results in a small loss of efficiency ([37]). On a positive note, this randomness makes the system highly opportunistic, which may be more natural than forcing some contrived control mechanisms.

In ART, static priority values can be attached to rules to control their position on the agenda. All contradiction rules are assigned the maximum priority so that they are detected as early as possible. This scheme avoids useless work in worthless contexts.

Also there is a difference in the structure of the problem-solver nodes in ART and the ATMS. Both systems have a datum and a label associated with each node. The ATMS also associates *justifications* with each node. ART attaches justification only if explicitly instructed to do so. This justifications are used to thread dependency links so that when a node is deleted, all the nodes that depend on it are automatically deleted. The MCTMS does not employ justifications at all.

4 Description of approach and some results

Our task is to detect buildings in aerial images. We restrict the shape of the buildings to be a solid bound by horizontal and vertical planes only. Further, adjacent planes must be mutually orthogonal. This is the same as the building model used in [6]. When a building that obeys this model is imaged from a point that is far away compared to the focal length of the camera (as is typical in aerial images), orthography can be assumed. Under orthography, parallel lines in the world appear as parallel lines in the image. For example, all the vertical lines appear as parallel lines. If the viewpoint vector is known and also the ground plane normal, then the direction of these lines in the image can be calculated. This direction could be used to identify vertical edges of buildings. The lines joining different object-shadow vertex pairs in the image will be parallel, since they are parallel in 3-D (they are along the direction of illumination vector). This direction in the image can be calculated beforehand if the direction of the illumination vector is also known (in addition to the viewpoint vector). Assuming that shadows are cast on horizontal planes (the ground plane or a plane that is parallel to it), object edges and their corresponding shadow edges will be parallel in 3-D. Consequently, object-shadow edge pairs will be parallel in 2-D. Since the viewpoint is close to directly overhead (the viewpoint vector is almost normal to the ground plane), adjacent horizontal edges of a building in the image are approximately perpendicular in the image.

In real images some of the object edges could be partially occluded; for example, by trees. Edges that are adjacent to shadow regions have a strong contrast. However, the contrast of other edges could be very poor, especially if the viewpoint is such that the edge is across two very similarly colored and oriented planes in the image. The current crop of low-level modules are far from perfect. The high level module has to deal with an input of fragmented and incomplete lines. The situation is further complicated by the fact that no real world building is a perfect rendition of the building model used. Surface markings, windows etc., could further obscure matters. Improvement, in the resolution of the model is difficult, mainly because there are too many possibilities. We have to stop at certain granularity. Given this uncertainty and ambiguity, clearly, there is a search space involved. The task of the high level module is to somehow synthesize buildings, using our knowledge of imaging geometry, shadow geometry and knowledge of the structure of buildings.

We have chosen to cast the problem of synthesizing buildings as a hierarchical composition process. This step is important. This reduces system complexity, permits reusability of parts and increases system robustness by permitting accumulation of intermediate evidence ([34]). First, object vertices are identified. Vertices are linked to form edges. Edges are linked to form rings. Closed rings correspond to object faces. Faces are grouped to form blocks. Blocks are glued together to form buildings. Missing features are hypothesized and if possible, confirmed. Each of the element in this hierarchical network is represented by a frame. Relationships between the elements are also represented by frames. The frame is the uniform representational unit used. An object oriented programming paradigm is used. Operations over objects are performed by passing *messages* to the objects. *Demons* defined over some frame slots are triggered asynchronously by certain operations over the slots, and usually take care of side effects of operations. This object oriented approach permits the system designer to work at a higher level of programming abstraction, making the system easily modifiable and extensible. Figure 12 shows part of a typical frame network.

Control of the problem solving process is primarily embedded in a set of production rules that perform a search over a search space using a multiple context truth maintenance system. The production rules are modular chunks of knowledge that are reasonably independent of each other. So extending the knowledge base is a simple matter of adding more rules (in real systems, there is certain amount of rule interaction. Our objective has been to minimize rule interaction to make the system as markedly modular as is permitted by the dynamics of the problem solving process). Figure 11 gives a schematic of the system.

4.1 System Implementation

The line segments produced by the low level module are asserted into the database as *instances-of* a generic line-segment frame. The line segments have essential slots for the end points and average contrast. The equation of the line can be written as

$$\cos\theta \cdot x + \sin\theta \cdot y + \delta = 0.$$

The coordinate axes are the x and y directions. θ is the anti-clockwise angle of the line from the x-axis and $\mid \delta \mid$ is the perpendicular distance of the line from the origin. Redundant slots, cache the values of $\cos\theta$, $\sin\theta$, θ and δ. This permits the rapid computation of relations between lines. For example, the intersection point of two lines

$$\cos\alpha \cdot x + \sin\alpha \cdot y + \nu = 0$$

and

$$\cos\beta \cdot x + \sin\beta \cdot y + \mu = 0,$$

is quickly computed as

$$(\frac{\mu \cdot \sin\alpha - \nu \cdot \sin\beta}{\cos\alpha \cdot \sin\beta - \sin\alpha \cdot \cos\beta}, \frac{\nu \cdot \cos\beta - \mu \cdot \cos\alpha}{\cos\alpha \cdot \sin\beta - \sin\alpha \cdot \cos\beta}).$$

When a non redundant slot is altered, demons attached to it can update the values of the redundant slots.

Each line is indexed into histogram of θ values as also the δ values. The histogram step size can be fixed at run time by the user. The histogram of θ values helps in a quick search for parallel line segments (parallel line segments will be in the same or adjacent histogram buckets of θ). All pairs of perpendicular lines can be quickly detected by examining pairs of lines from appropriately spaced θ–histogram buckets (depending on the step size of the divisions). Potential colinear lines can be detected quickly using the histogram of δ values (colinear lines are at the same perpendicular distance from the origin. So they will be in the same or adjacent histogram buckets of δ). The other alternative to histogramming is to compare every line with every other line, an unattractive and computationally intensive $O(n^2)$ operation.

Potential vertices are detected in this image by looking for coincident and mutually perpendicular lines in the image. The vertices detected in the image in Figure 3 are shown in Figure 13 as black circles. The arrows pointing away from the vertices are *limbs.* Limbs are vectors that originate at a vertex and point along the direction of one of the lines forming the vertex. A limb is also represented by a frame. A limb-frame is very similar to a line-segment-frame. It is also indexed into histograms for fast comparisons. The limb and the vertex it belongs to are connected with pointers in both directions (the pointers are also frames). If two limbs are coincident and are facing each other, then their corresponding vertices can potentially form an edge. Such limbs are called *compatible limbs.* All pairs of compatible limbs are identified. This is a robust technique for detecting object edges, because object edges are frequently fragmented into smaller segments in the input to the higher level module, for the reasons cited elsewhere in this section. A possible object edge connection exits between the vertices if there is sufficient low level evidence of a line extending between the two vertices. To determine this, we examine a narrow rectangular area along the edge and make sure that there are no large gaps in the low level line evidence. Then an *assumption* that the vertices are connected is asserted into the database. The MCTMS will work with this base set of assumptions.

The set of vertices and their assumed connections form a graph. This problem graph has to be searched for closed edge rings. Additional edge *assumptions* need to be made to close partially closed edge-rings. Any scheme (eg: depth-first, best-first, breadth-first) can be used to perform this search. This problem is potentially exponential. One way of limiting search time is to *assume* as edges only those vertex connections that have strong low level evidence. Also when edge-rings are linked, knowledge of the structure of building roofs (adjacent edges are orthogonal) is used. Further efficiency is gained by utilizing the MCTMS to perform an efficient search of the problem graph. The advantages of an MCTMS aided search over a node-at-a-time search schemes was detailed in Section 3. We try to recapture the list of advantages again.

- There is virtually no rediscovery of information when using the MCTMS. There is no such guarantee when using a simple search scheme.
- If a closed-edge ring is confirmed as a roof, then all the underlying object edge assumptions are correct. This information must be transferred to other points of the search space. This can be done by using the *confirm* function described in 3.5. This collapses the confirmed context into the root. The information is then available in all other contexts through inheritance. There is no such simple way to transmit information in simple node-at-a-time search schemes.
- If a edge-ring is in error (for example, self intersecting), then the environment of that context is a nogood. This is marked as such in the context tree and no descendants of this path are tried again. A simple node-at-time search procedure could rediscover this contradiction from different paths.
- The MCTMS is completely opportunistic. This fits in well with the paradigm of rule-based programming. Attempts to force a particular ordering of search in rule based schemes is not natural. Advantages of modularity and opportunism can be easily lost.
- It is virtually impossible to compare across contexts in simple search scheme. In an MCTMS approach this is easily done. For example, if there are two edge-rings that mutually intersect, then the merge of their contexts will be contradicted. Subsequently, if sufficient shadow evidence is found for one ring then it's context can be *confirmed*. The MCTMS automatically contradicts the competing context and stores it's environment in a nogood database.

4.2 Future Work

Our system is still in a preliminary stage of construction. Currently we can compose edge-rings and detect closed edge-rings. The following paragraphs detail some of the proposals for completing the system.

- The initial focus will be on closing each edge-ring detected in the bottom-up organization. A detailed search will be made in the database along the *limbs* of the end vertices of the edge-rings, to look for vertices that can potentially continue this edge-ring with relaxed thresholds. If there is evidence of any such vertex then an edge between the two vertices can be *assumed* and this will participate in the edge-ring formation process. This would correspond to a top-down organization. The magnitude of the thresholds used initially will determine the degree to which the system works bottom-up or top-down.

To compare competing chains (in competing contexts) the following set of intelligent heuristics can be used.

1. If a chain is complete and closed then *confirm* the context it is in and *contradict* all competing contexts.
2. Prefer the longer chain. If the chains are of the same length, then prefer the one with the larger *cover*.
3. Use shadows to confirm roof hypotheses.

- Shadow analysis has to be smoothly integrated into the system. Currently we are looking into Shafer's ([39]) work on shadows cast by polyhedra with two visible surfaces and a self shadowed surface. The shadows could be cast on other polyhedra. This is a relatively complex problem. However, our building model uses a restricted class of polyhedra. This could simplify our task.

- Efficiency issues in dealing with a MCTMS have to be addressed in greater detail. The comparison in Section 3.5 between ART's MCTMS and de Kleer's ATMS needs to be further refined. In [40], Provan addresses the complexity issues associated with the use of the ATMS in scene analysis. This could serve as the starting point for a more detailed analysis.

- The issue of detecting different kinds of objects (eg: planes, roads, buildings) needs to be addressed.

  Frame based knowledge representation will be especially useful for modeling different classes of objects. ACRONYM has shown how powerful frames can be in modeling generic classes of objects. Frame based modeling can also be extended to scenes (eg: an airport scene). Default values of the scene frame slots could serve as expected scene organization. Minsky originally suggested that frames be used to represent prototypical situations. However, the guidelines on how exactly to go about this process are at best vague. We plan to closely examine this issue.

  Will an ATMS-type search through the alternatives be appropriate in this task too? This issue needs to be investigated.

- Another important extension would be to incorporate into the system, an ability to incrementally update the scene model with each new view. The 3D-MOSAIC [4] is an example of a system that can do this. However, it uses a relaxation method to match the new model to the old and to resolve inconsistencies. This is a fertile area for the application of a truth maintenance approach. The reason is that the process of assimilating new information includes a process of belief revision where new information can confirm or contradict old information. TMSs are specifically designed to efficiently address this belief revision process. Change detection will be a part of this belief revision process.

- Stereo matching is another issue we intend to investigate. Different techniques have been used to match structures across a stereo image pair (for example, the 3D-MOSAIC uses a beam search to match junction across two images). An appropriate approach to this problem seems to be to use some kind of a constraint mechanism to prune the number of matches and then use an explicit search technique (possibly using the MCTMS) to intelligently match the remaining structures. This approach would be semantically similar to the *plan-generate-test* paradigm used in DENDRAL ([41]), a system for inferring structures of organic compounds based on nuclear magnetic resonance (NMR) data and mass spectrograms. The *plan* phase uses constraint-satisfaction techniques to create much reduced lists of recommended and contraindicated substructures. The *generate-and-test* phase then uses these lists to explore only a limited number of structures.

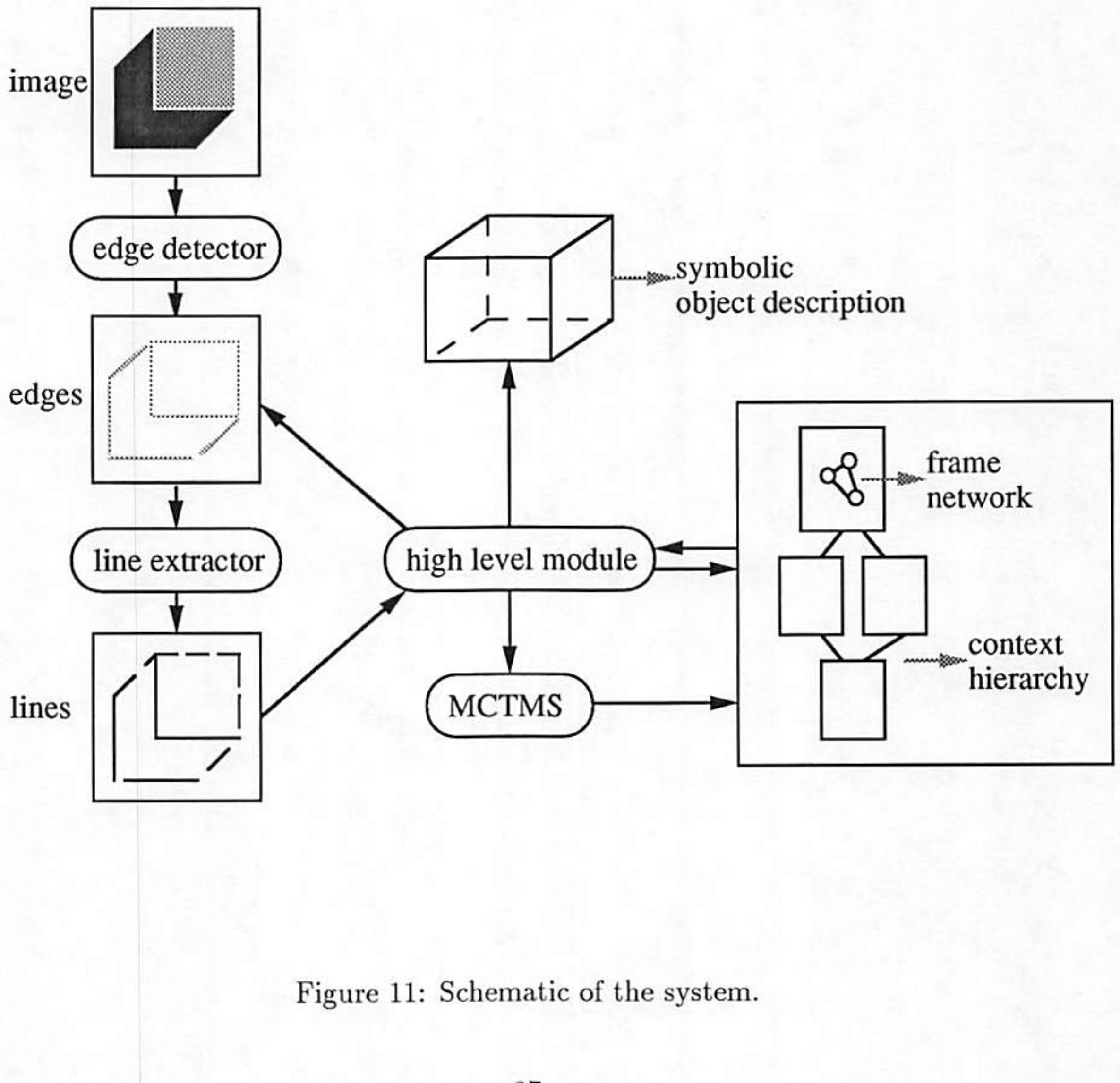
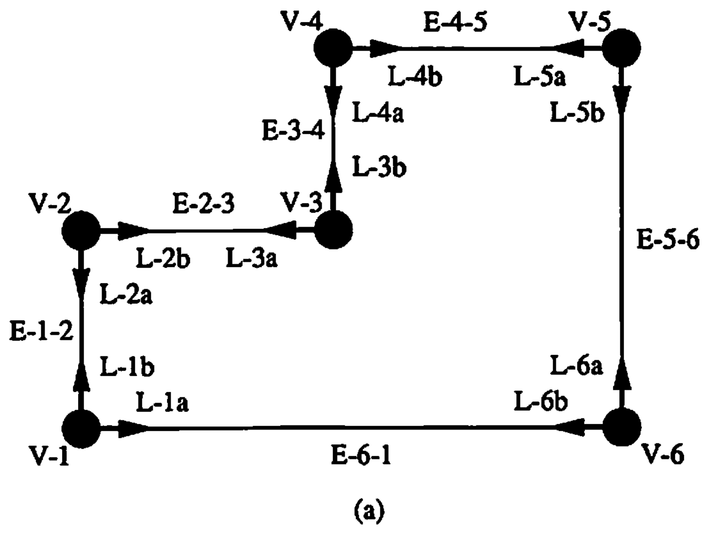


Figure 11: Schematic of the system.



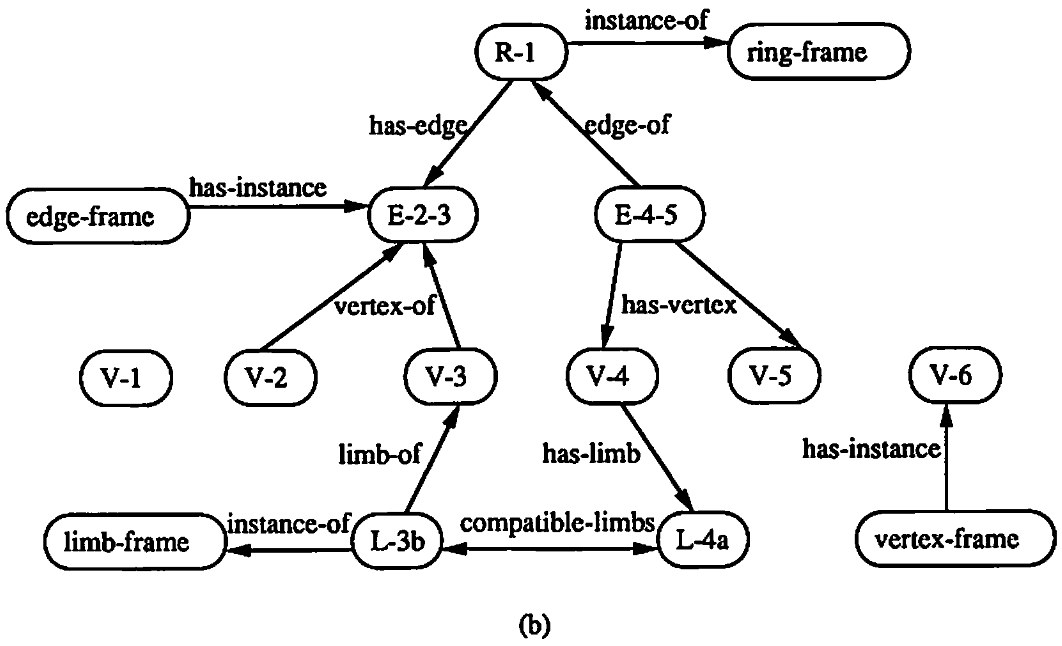


Figure 12: Illustration of the frame network. (a) A simple roof. (b) Part of the frame network of (a).

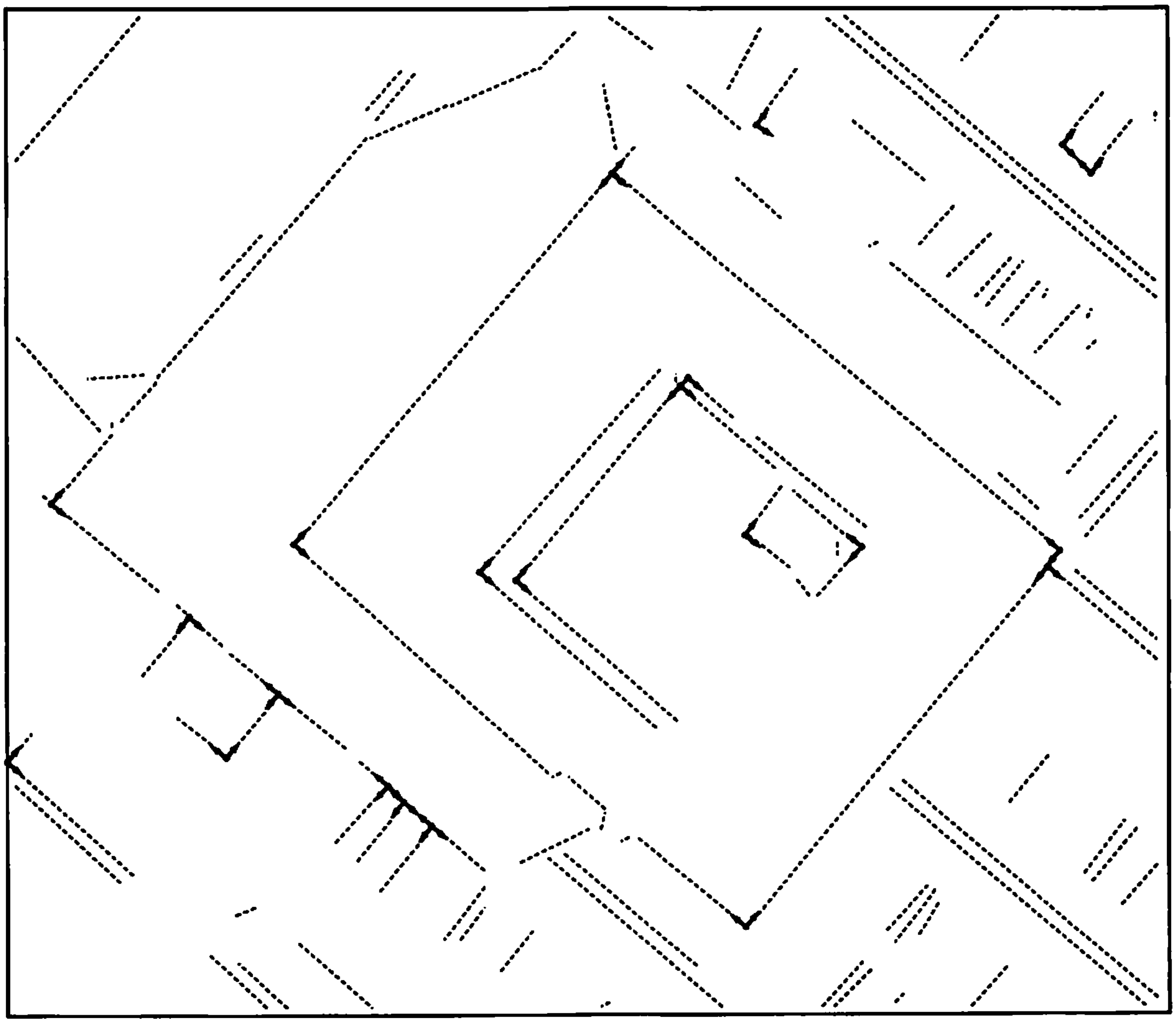


Figure 13: Vertices and Limbs.

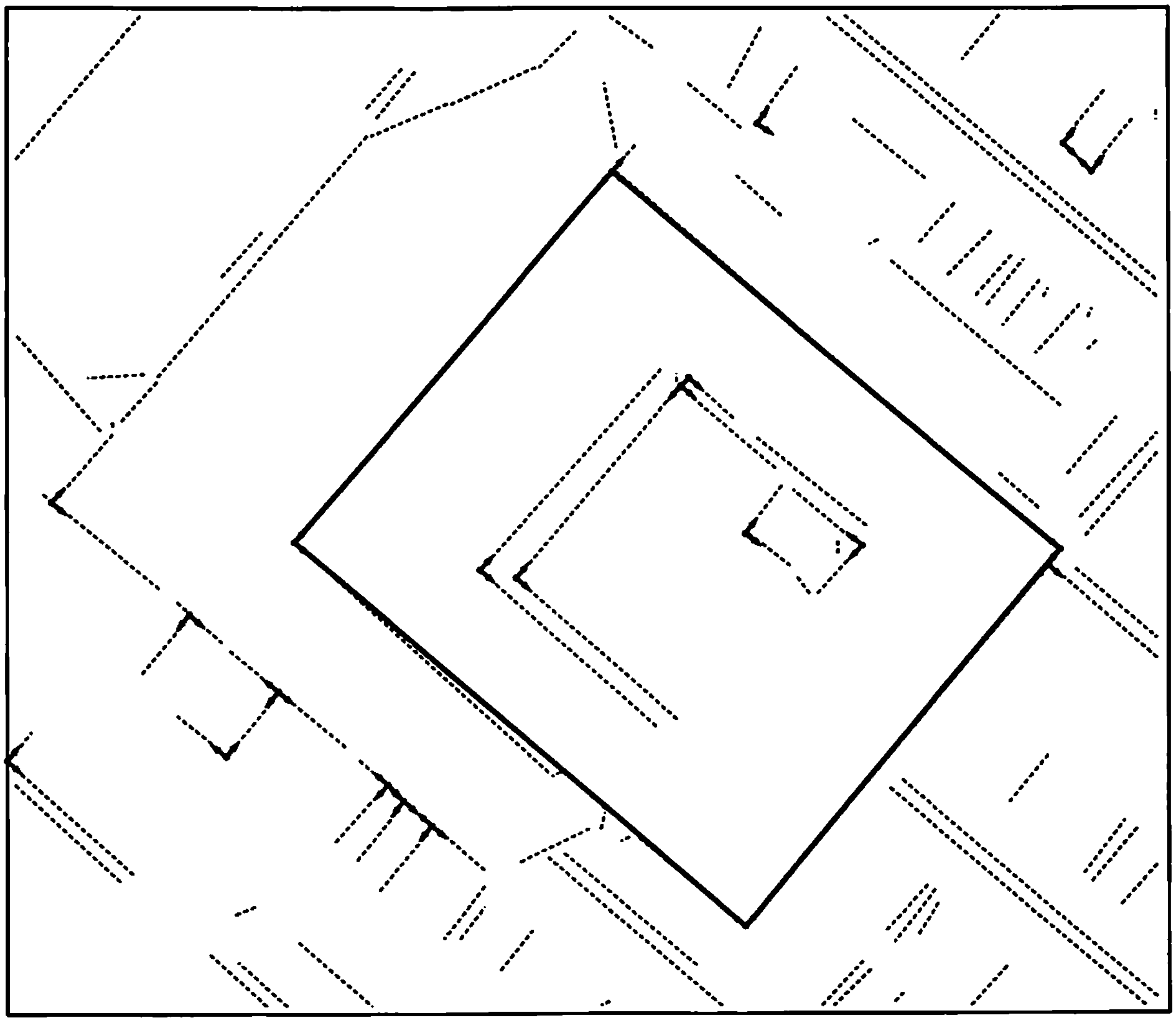


Figure 14: A closed ring.

5 Overview of Related Work

5.1 3-D MOSAIC

The 3-D MOSAIC system can reconstruct 3-D scenes from a sequence of images obtained from multiple viewpoints. The system effectively has a low-level, an intermediate level and a high level module. The low level module is a line extractor. The intermediate level module can use one of two components - a stereoscopic component and a monocular component, depending on the type of data available. Both components produce sparse wire frame data organized into a hierarchical graph structure, called a *structure graph*. The high level module merges this data into the current scene model (incremental reconstruction) and updates the structure graph using domain knowledge to derive full surface based descriptions. There is no feedback between the three modules. The system is demonstrated on the task of detecting buildings with horizontal and vertical faces.

The monocular component of the intermediate level derives 3-D wire-frames from the input lines in an entirely algorithmic manner using task-specific assumptions. The stereoscopic component derives junctions in the stereoscopic pair and uses a beam search to derive a unique set of junction matches. These junctions are then triangulated and further processing finally results in a hierarchical graph structure.

The structure graph is a graph structure that is geared for belief revision. Modifications (like addition and deletion) are propagated along the graph based on *part-of* and *constrains* links. Detailed techniques are described on how to deal with geometric and topological modifications. The concept of the structure graph with revision capabilities can be construed as a domain specific specialization of Doyle's ([20]) TMS. Dependencies are implicit in the form of two links; *part-of* and *constrains*. Dependencies cannot be cached, because when a link is deleted so is the dependency information. Rediscoveries of inferences and contradictions may not be avoidable. Consequently, much of the power of the TMS is lost in this specialized structure. Dependency-directed backtracking is not possible. This technique permits fast modifications, but coherence and correctness are partially sacrificed for the sake of speed.

The high level module can update the structure graph into a full surface based graph description based on knowledge of planar faced objects and knowledge of urban scenes (missing elements are hypothesized during updating). Before that, the high level module has to reconcile the new wire frame data from the current scene to the previous scene model. First a matching is done using a relaxation method. Consistent matches are averaged. If a hypothesized element has an inconsistent match it is deleted, propagating the effects along the structure graph as described in the last paragraph.

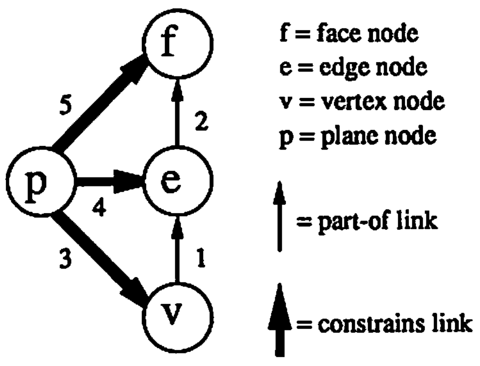


Figure 15: An example from the 3-D MOSAIC system.

For an example of incoherence in the system, consider this example taken from [4]. In Figure 15, suppose link 4 is deleted (i.e. the assertion "p constrains e" is deleted). Then either of link 1 or 3 could be deleted. A heuristic that *part-of* links should dominate *constrains* link is used to delete link 3. Many such local heuristics are used to effectively algorithmically come to a quiescent state.

5.2 SIGMA

In [32], Hwang et al. describe a general framework based on integration of hypotheses for building vision systems. They also describe SIGMA, a specific implementation using this framework for detecting buildings in suburban scenes.

A general system is divided into two parts, a low level vision system (LLVS) and a high level vision system (HLVS). Objects are represented by *frames.* Relations between objects are represented by *rules* and *links* which are stored in frames. CAN-BE and AKO (a kind of) links are used to arrange the frames in a hierarchy. Rules have three parts:

$< control - condition >$

$< hypothesis >$

$< action >$

The task of the LLVS is to generate all possible instantiations given a *primitive* < *hypothesis* > (a *primitive* frame is one that the LLVS is capable of detecting; for example, rectangular blobs) as a goal. An initial set of *primitive* hypotheses is provided to the LLVS to start the system working. The HLVS iterates the following steps.

1. Examines the $< control - condition >$ of each rule of the frame instantiations in

the database and if satisfied generates the $< hypothesis >$. Then, clusters related hypotheses into situation sets and arranges them in a situation lattice.

2. Focuses attention on the strongest cluster and constructs a *composite* hypothesis by utilizing procedures stored in the corresponding generic frame (typically procedures are simple union or intersection).

3. Attempts to verify the *composite* hypothesis by looking for already existing frame instances. If this fails and the *composite* hypothesis is actually *primitive* then the HLVS directs the LLVS to generate potential matching instantiations. If the hypothesis is not *primitive*, then it is recursively decomposed into sub-hypotheses, using *decomposition strategy rules* stored in the corresponding frame.

4. Executes the $< action >$ parts of the rules if the $< hypothesis >$ is verified. Typical actions include creating new instances, unifying instances, updating or removing hypothesis etc.

Conflicting instances coexist in the database. An interactive interface, called the Query Answering Module (QAM) is provided for the human user to somehow manually choose the "best interpretations". The QAM effectively side-steps the entire important issue of automatic resolution of conflicting interpretations.

The framework proposed is nice and elegant. However, SIGMA (a specific implementation of this framework in a simple 2-D domain) is a weak illustration of this framework. The frame hierarchy is shallow and the examples shown are simple. The center piece of this work, hypothesis integration, is simple union or intersection operations (mainly because slots can only be constrained to lie in an interval. Brooks' ([3]) work on manipulating more complex constraints on slot values can be potentially employed here).

5.3 3-D FORM

The 3-D FORM system of Walker et al., described in [31], is a 3-D photo-interpretation system that is based on frames. Both representation and reasoning capabilities are fully integrated into frames. Order of process is largely controlled by the order of slot-access. Simple frames are first instantiated based on the input data. A hierarchical grouping process is used to group lines into faces and faces into objects (vision researchers have consistently found it to be advantageous to depict a hierarchy and break down a complex task into a more organized and simpler parts. The structure graph used in the 3-D MOSAIC is also a hierarchical structure. We have used a hierarchical organization of object frames in our system also). Relationships between the object subframes are also instantiated. After this initial grouping, all objects are of the most general type; for example, 3-D-OBJECT. Objects are then specialized along an IS-A hierarchy as much as possible and if needed, a slot by slot matching to prototype frame slots can be attempted, using the object slot constraints to resolve local ambiguities. Constraints (associated with each frame) are then evaluated to provide hypotheses for missing slots of objects.

The system is primarily a high level module. The input data is hand generated 3-D wireframes. The wireframes have virtually no ambiguity (for instance, the lines in the wire frame are cleanly connected at the vertices without ambiguity. Also, the wire frames are already segmented as belonging to separate and distinct objects). These wireframes are interpreted to detect a much restricted class of buildings (polygonal roofs, horizontal and vertical faces). The task of getting such a high-level data is not an easy one. The issue of implementing the low-level and the intermediate-level modules to automatically generate this data is not addressed.

Control is completely local to frames. Consequently, no coherent, global and complete search paradigms are possible. One local form of error recovery tried was to store recovery procedures in *recovery facets* of frames. This requires painstaking identification of all possible situations of constraint failures and defining a procedure for error recovery for each of these cases. The recovery procedures are at the lowest level. No recovery procedures are listed at the higher levels (for example, what will happen if the lines are organized into improper wireframes?). At the time of this report, this system was still in a preliminary stage of implementation.

5.4 CANC

In [5], Mohan and Nevatia make a case for giving a role to the process of perceptual organization in vision systems. Primitive image elements generated by the low-level module are grouped into *collated features* which encode the structural interrelationships between the elements. The kind of structural relationships chosen are those that are invariant to viewpoint. They should also be significant (should correspond to object structures) and should be useful to the other visual processes to work with. An implementation is demonstrated over a limited visual domain (much like ours) of buildings that are compositions of rectangles. The collated features in this domain are defined to be lines, parallels, Us and rectangles organized in a hierarchy. These features are arranged in a network. The nodes are the collated features. Arcs with positive weights link mutually supportive collations (like those in a hierarchy) and arcs with negative weights represent mutually competitive collations (for example; two overlapping rectangles). This can be characterized as a constraint satisfaction network, which is a subset of the PDP models discussed in [42]. The potential at each node is updated in a relaxation scheme till a locally optimal state, in which as many as possible of the constraints are satisfied, is reached. Globally optimal solution is not guaranteed (this would require the use of stochastic elements and a time consuming process of annealing). This is essentially a connectionist solution to an intermediate level vision problem. The arguments detailed in 3.3 are valid here. The weights are manually assigned and there is no "learning" scheme.

The high level collations (rectangles) are used to guide processes like stereo and also object recognition. Rectangles are matched across two images for the stereo procedure. The process of object recognition consists of combining rectangles based on *subsumption*, *merger-compatibility*, *occlusion* and *incompatibility*. Every pair of rectangles are tested in the first sweep. New rectangles formed, participate in the next sweep and this process continues till there are no new structures generated. This is a rather expensive, iterative process. The final structures are then assigned heights from the disparity information previously obtained by stereo.

5.5 Huertas and Nevatia

Huertas and Nevatia present a system for detecting buildings in aerial images in [6]. Buildings are detected by first looking for structures in the image that are compositions of rectangles, hypothesizing them as roofs of buildings and then searching for shadow evidence to confirm the hypotheses. First, lines and corners in the image are detected. The corners then are labelled as object or shadow corners using knowledge of the direction of illumination and identifying shadow regions of the corners based on the average gray level of shadows in the image. Object corners are linked in chains to form object boundaries. If a chain is not closed, looser tolerances are used to search for corners to continue the chain.

The system described in Section 4 bears some similarities with this system. However, this system has a strong algorithmic flavor. For example, consider the case of detecting corners. A 5×5 window at the end point of each line segment is searched in a spiral fashion, stopping at the first potential corner that is detected. Alternatives are not considered. When matching object corners to shadow corners, in case of a multiple match, the closest corner is chosen (it is assumed that corners are correctly identified as object or shadow corners). Again, alternatives are not considered. The authors describe a mechanism for backtracking in one particular situation faced when tracing object boundaries. Sometimes a shadow corner may be reached erroneously when tracing object boundaries. In this case backtracking is done to the last visited junction in the chain where there was a change in brightness in the corners in the chain. This is much like the *recovery facets* used in the 3-D

FORM (summarized in 5.3). These special case procedures define mechanisms for local error recovery for specific cases. However, the issue of dealing with alternatives in general cases is not addressed.

5.6 SPAM

McKeown et al. present a rule-based system for the photo-interpretation of airport images in [2]. Elements in the database are organized in a hierarchy. Regions identified by a region extractor are first interpreted as *fragments* (eg: terminals, roads etc.). For each seed *fragment* (terminal, roads, hangars and runways), the list of fragments with which it is "consistent" is traced to determine whether there is sufficient evidence for a *functional area* (eg: runway functional area, terminal functional area etc.) to be formed. A mutually consistent collection of *functional areas* form a *model* (eg: an airport model). Top down analysis is possible. Predictions can be made based on incomplete *functional area* cliques and rules can invoke image analysis tools to go look for appropriate regions to satisfy the predictions.

Conflicting interpretations coexist in the database. Much of the model evaluation phase of the interpretation is organized to resolve inconsistencies between overlapping functional areas. One way to resolve conflicts is to choose the interpretation with the higher confidence value.

The system is implemented in OPS-5, a production system language that is one generation removed from the current state-of-the-art knowledge engineering tools. If OPS-5 were augmented with a context mechanism, truth maintenance capabilities, a frame based representational language and more, the resulting language could be ART (our implementation language).

5.7 ACRONYM

ACRONYM ([3]) is a very general and influential vision system. Frames are used to represent objects with sub-parts in an *object graph*. Slots can be filled with *quantifiers*. Generic classes of models can be specified by supplying a set of algebraic inequalities over these quantifiers. Each set is encapsulated as a node in a *restriction graph*. Directed arcs in the *restriction graph* specify sub-class inclusion. A complex geometric reasoning system (built using production rules) generates a *prediction graph*. The nodes are predictions of image features and the arcs are relations that must hold between them in the image. Guided by the predictions, an *observation graph* is generated from the image by a low-level module.

The interpreter then attempts to match maximal subgraphs of the *observation graph* to subgraphs of the *prediction graph* using a relaxation algorithm. Each such match is an *interpretation graph.* Interpretation nodes relate prediction nodes to observation nodes. Associated with a prediction node are instructions on how to use image measurements to generate back constraints. These instructions are placed there by the geometric reasoning system. The generated back constraints are placed in a new restriction node which is more restrictive (and therefore inherits constraints) than the one associated with the prediction node. If the set of constraints are unsatisfiable, then the interpretation node can be eliminated from further consideration.

At this stage the interpreter now has hypothesized connected components of the *interpretation graph.* A combinatorial search is carried out to find consistent connected components. A number of interpretation graphs can be hypothesized. A single interpretation is somehow synthesized from them.

5.8 SCERPO

In [15], Lowe presents a system for recognizing known 3-D objects in an image. The object models consist of a set of 3-D line segments. Generic models cannot be defined. Initially, perceptually significant and viewpoint invariant groupings are generated from the line segments of the object models (for example; groupings based on colinearity, proximity and parallelism of line segments are perceptually significant and viewpoint invariant). More complex viewpoint-invariant structures can be formed by combining groupings that share line segments. Similar groupings into complex structures is also done in a bottom-up fashion from the line segments in the image. A matching algorithm then matches these structures of the object model with the structures generated from the image. This is used to constrain the 3-D position of the model, which in turn leads to further predictions on the locations of the projections of the model line segments in the image. Newton's technique is used to iterate and converge to a viewpoint, if possible. When an object is detected (by convergence), the line segments that constitute the object are not considered for any other groupings.

This work is similar to ACRONYM. In both, the image primitives are grouped based on viewpoint invariance. Lowe is able to compute measures of significance for the observed viewpoint invariant relations. These significance values are used to focus on stronger groupings first. Lowe does not delve very deep into the issues of matching predicted features with observed features. ACRONYM uses a graph matching procedure. Because

SCERPO has precise models of objects, a numerical method suffices to extract objects. Since ACRONYM represents generic classes of objects, based on algebraic inequalities over quantifiers, a complex symbolic equation solver is used to place suboptimal bounds. However, this in no way detracts from the importance of this work, which is very general and influential in scope.

5.9 Nagao and Matsuyama

In [1], Nagao and Matsuyama use a blackboard system ([43, 44]) to interpret multi-spectral high-altitude aerial images. The image is first segmented into *elementary regions* based on multi-spectral properties. Each elementary region has a unique label number. On a *label picture*, each pixel has the label number of the elementary region it belongs to. This is a useful data structure, because it permits manipulations at the pixel level, like shrinking and expanding of regions (the label image used in our line extractor defined in Section 2 is a similar data structure). Several basic properties of each elementary region, such as the average gray level in each spectral band, shape features from the minimum bounding rectangle, coordinates of centroid etc., are stored in a *property table* on the blackboard. Based on the properties stored in the property table, a group of mutually independent *characteristic region extractors* extract characteristic regions. On the blackboard, each characteristic region is represented as a set of elementary regions. The system is equipped with *knowledge sources* called the *object detection subsystems* (ODSS). Data-driven ODSSs focus attention on specific local areas by combining several characteristic regions (through boolean operations) and check for the existence of specific objects by consulting the knowledge stored in the subsystem. Model-driven ODSSs use spectral properties and spatial relationships with the regions already classified to classify new regions. There is no scheduling of knowledge sources as in HEARSAY-II ([45]). All the ODSSs that are triggered are fired in parallel. A region could possibly get more than one label (eg: forest and crop) from the ODSSs. Such conflicts among the ODSSs are resolved by re-examining low-level data in a top-down fashion. For example, the value "irregular shaped" in the recognition-status field of candidate regions in the property table triggers segmentation processes which attempt merge-split techniques to get more regular shapes. Dependency pointers are maintained so that when an interpretation is rejected, all the interpretations that depended on it are also rejected.

5.10 The VISIONS project

The VISIONS project at the University of Massachussets has been in progress for more than a decade. This is an evolving general purpose vision system. In [33], Hanson and Riseman outline the basic strategy in building the VISIONS system. A priori knowledge of the scene is stored in a schema network in the long term memory (LTM). During the interpretation of an image, a network of schema instances is created as a description of the image in the short term memory (STM). Declarative knowledge is stored in schemas. The problem space is divided into a hierarchy of representation levels forming a black board. Procedural knowledge is encoded in *knowledge sources*, which take data from one level and act at the same or different levels. The interpretation is opportunistic and can be either top-down or bottom-up. A hierarchical modular control strategy is described. It essentially acts in a *focus, expand and verify cycle.*

This system is similar to the HEARSAY-II ([45]) speech understanding system. In HEARSAY-II all the partial, competing models are stored in a single network. This allows easy access to competing partial models. However, the history of development of each partial model is not maintained. In the VISIONS system, CONNIVER ([46]) style *contexts* are used to store each partial model. The context tree maintains the search space history. This aids in debugging and in making control decisions. However as in most *single-state* context systems, simultaneous access to competing partial models is difficult. The MCTMS approach we use in our system, described in Section 4, allows easy access to competing models in a context framework. Effectively, it combines the advantages of the search space representations of HEARSAY-II and VISIONS.

The VISIONS system has evolved over the years ([47]). The trend in the evolution is to distribute control within schemas, making each schema completely responsible for itself. There is no longer a need for a central control engine (like the hierarchical modular control system mentioned previously). *Interpretation strategies* associated with each schema provide the control. Schemas communicate indirectly by posting *object hypotheses* on a blackboard.

Weymouth in [27] developed a schema-based interpretation system to be embedded into the framework of the VISIONS system. He advocates more direct communication between schemas through the establishment of communication links. In the VISIONS system the LTM is a collection of all the object representations and control information. In Weymouth's implementation of the interpretation system, a schema network with parallel distributed control is used in the LTM. Each schemas represents an object to be recognized and the methods for recognizing it (stored as *interpretation strategies*). In data-driven activation, any event from the "list of important events" (associated with each schema) can activate the schema. In goal-driven instantiation, a goal is generated by some active schema. A schema that can potentially satisfy this goal is activated and attaches a schema instance through a communication link. Messages can be passed across these links facilitating direct communication (schemas can also communicate indirectly through shared data. But this form of communication is rarely employed). A schema hypothesis is created if there is enough confidence that the goal will be satisfied. A schema network is thus created through the interpretation process. It is hoped that redundant information distributed in schemas (as a large number of *interpretation strategies*), will somehow "cut through" the noisy and error ridden low-level data.

A system called GOLDIE was designed by Kohl ([48]) to provide intelligent intermediate level control within the frame work of the VISIONS system. GOLDIE mediates the interaction between high level interpretation strategies and the low level segmentation operations. In the data-driven mode, GOLDIE functions as a segmentation system that chooses between a variety of low-level algorithms and features, based on the nature of the data. In the goal-driven mode, high level processes post *goals* on a blackboard. This results in the creation of an intermediate level process which is an instantiation of a schema control strategy for realizing that goal. GOLDIE extends the concept of schema-directed control (that permeates the current VISIONS system) to the low and intermediate level processes.

5.11 COBIUS

COBIUS ([49]) performs constraint-based interpretation of aerial images. Frame hierarchies are used to represent objects as well as constraints. Production rules are used to formulate hypotheses and manipulate constraints. Primitive and complex constraints are arranged in a PART-OF hierarchy. Combinatorial complexity of graph matching is avoided by composing matched primitive constraint instances into more complex constraint instances and predicting and verifying the missing primitive constraints. Control knowledge is explicitly coded as a mixture of frames and rules. This system is implemented in ART. The blackboard model of problem solving ([43, 44]) is used.

5.12 VICTORS

VICTORS ([8]) used the ATMS (a multiple context truth maintenance system) to solve the simple problem of identifying puppets from an input of rectangles. In our work, we are using an MCTMS on real images and for a much more complex vision task.

6 Conclusions

We are in the process of building a system for the intelligent interpretation of aerial images. The domain of interpretation is restricted (as in [5, 6]) to the identification of buildings that are compositions of rectangular parallelopipeds. However, we address some fundamental issues in the application of artificial intelligence methods to vision systems in general. Though the issue of knowledge representation seems to have been adequately dealt with in vision systems, surprisingly enough, the issue of coherent and efficient search has not been addressed adequately. We tried to provide the motivation for a coherent search strategy based on a MCTMS and an object-oriented frame-based knowledge representation approach. Some of the work reported here is preliminary in nature. We are currently working on completing the system and putting it in proper perspective with respect to other vision systems.

7 Acknowledgements

Special thanks to Mr. Andres Huertas for providing the images and help with the Nevatia-Babu LINEAR program.

References

[1] M. Nagao and T. Matsuyama, *A Structural Analysis of Complex Aerial Photographs*, Plenum, New York, 1980.

[2] D. M. Mckeown, W. A. Harvey, and J. McDermott, Rule-based interpretation of aerial imagery, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-8, pp. 570–585, 1985.

[3] R. A. Brooks, Symbolic reasoning among 3-D models and 2-D images, *Artificial Intelligence*, 17, pp. 285–348, 1981, Special volume on computer vision.

[4] M. Herman and T. Kanade, Incremental reconstruction of 3D scenes from multiple, complex images, *Artificial Intelligence*, 30, pp. 289–341, 1986.

[5] R. Mohan and R. Nevatia, Perceptual grouping for the detection and description of structures in aerial images, In *Proceedings DARPA Image Understanding Workshop*, pp. 512–526, Cambridge, Massachussets, April 1988.

[6] A. Huertas and R. Nevatia, Detecting Buildings in Aerial Images, *Computer Vision, Graphics and Image Processing*, 41, pp. 131–152, 1988.

[7] E. Rich, *Artificial Intelligence*, McGraw-Hill, New York, 1983.

[8] G. M. Provan, Model based object recognition - A truth maintenance approach, In *Proceedings Fourth IEEE Conference on Artificial Intelligence Applications*, pp. 230–235, San Diego, California, March 1988.

[9] J. de Kleer, An assumption-based TMS, *Artificial Intelligence*, 28, pp. 127–162, 1986.

[10] J. de Kleer, Problem solving with the ATMS, *Artificial Intelligence*, 28, pp. 197–224, 1986.

[11] R. Nevatia and K. R. Babu, Linear feature extraction and description, *Computer Graphics and Image Processing*, 13, pp. 257–269, 1980.

[12] J. B. Burns, A. R. Hanson, and E. M. Riseman, Extracting straight lines, *IEEE Trans. Patt. Anal. and Mach. Intell.*, PAMI-8, pp. 425–455, 1986.

[13] Y. T. Zhou, V. Venkateswar, and R. Chellappa, Edge detection and linear feature extraction using a 2-D random field model, *IEEE Trans. Patt. Anal. and Mach. Intell.*, PAMI-11, pp. 84–95, January 1989.

[14] J. F. Canny, A computational approach to edge detection, *IEEE Trans. Patt. Anal. and Mach. Intell.*, PAMI-8, pp. 679–698, 1986.

[15] D. G. Lowe, Three-dimensional object recognition from single two-dimensional images, *Artificial Intelligence*, 31, pp. 355–395, 1987.

[16] L. S. Davis, Shape matching using relaxation techniques, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–1, pp. 60–72, 1979.

[17] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, Scene labelling by relaxation operations, *IEEE Trans. on System, Man and Cybernetics*, SMC–6, pp. 420–433, 1976.

[18] D. Waltz, Understanding line drawings of scenes with shadows, In P. H. Winston, editor, *The Psychology of Computer Vision*, pp. 19–91, McGraw–Hill, New York, 1975.

[19] S. E. Fahlman and G. E. Hinton, Connectionist architectures for artificial intelligence, *IEEE computer*, pp. 100–109, January 1987.

[20] J. Doyle, A Truth Maintenance System, *Artificial Intelligence*, 12, pp. 231–272, 1979.

[21] R. M. Stallman and G. J. Sussman, Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis, *Artificial Intelligence*, 9, pp. 135–196, 1977.

[22] D. G. Bobrow and T. Winograd, An overview of KRL, a knowledge representation language, *Cognitive Science*, 1, pp. 3–46, 1977.

[23] R. Fikes and T. Keller, The role of frame–based representation in reasoning, *Communications of the ACM*, 28, pp. 904–920, 1985.

[24] M. Stefik and D. G. Bobrow, Object-oriented programming: Themes and variations, *AI Magazine*, 6(4), pp. 40–62, Winter 1986.

[25] M. Minsky, A framework for representing knowledge, In J. Haugeland, editor, *Mind Design*, pp. 95–128, The MIT Press, Cambridge, Massachusetts, 1981.

[26] M. A. Arbib, Artificial intelligence and brain theory: Unities and diversities, *Annals of Biomedical Engineering*, 3, pp. 238–274, 1975.

[27] T. E. Weymouth, *Using object descriptions in a schema network for machine vision*, PhD thesis, University of Massachusetts, 1986.

[28] M. A. Arbib, Levels of modeling of mechanisms of visually guided behavior, *Behavioral and Brain Sciences*, 10, pp. 407–465, 1987.

[29] P. J. Hayes, The logic of frames, In D. Metzing, editor, *Frame Conceptions and Text Understanding*, pp. 46–61, Walter de Gruyter and Co., Berlin, 1979.

[30] D. McDermott and J. Doyle, Non-monotonic logic-I, *Artificial Intelligence*, 13, pp. 41–72, 1980.

[31] E. L. Walker, M. Herman, and T. Kanade, A framework for representing and reasoning about three-dimensional objects for vision, *AI Magazine*, 9(2), pp. 47–58, Summer 1988.

[32] V. S–S Hwang, L. S. Davis, and T. Matsuyama, Hypothesis integration in image understanding systems, *Computer Vision, Graphics and Image Processing*, 36, pp. 321–371, 1986.

[33] A. R. Hanson and E. M. Riseman, VISIONS: A computer system for interpreting scenes, In A. R. Hanson and E. M. Riseman, editors, *Computer Vision Systems*, pp. 303–333, Academic Press, New York, 1978.

[34] B. Chandrasekharan, A. Goel, and D. Allemang, Connectionism and information-processing abstractions, *AI Magazine*, 9(4), pp. 24–34, Winter 1988.

[35] M. H. Richer, An evaluation of expert system development tools, *Expert Systems*, 3(3), pp. 166–183, 1986.

[36] W. Metrey, An assessment of tools for building large knowledge-based systems, *AI Magazine*, 8(4), pp. 81–89, Winter 1987.

[37] Inference Corporation, Los Angeles, *ART reference manual*, January 1987.

[38] C. L. Forgy, Rete: A fast algorithm for many pattern/many object pattern match problem, *Artificial Intelligence*, 19, pp. 17–37, 1982.

[39] S. A. Shafer, *Shadows and Silhouettes in Computer Vision*, Kluwer Academic Publishers, Boston, 1985.

[40] G. M. Provan, Complexity analysis of multiple context TMSs in scene representation, In *Proceedings AAAI*, pp. 173–177, 1987.

[41] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, *Applications of Arttificial Intelligence for Organic Chemistry: The Dendral Project*, McGraw-Hill, New York, 1980.

[42] J. L. McClelland, D. E. Rumelhart, and the PDP research group, editors, *Parallel Distributed Processing: Explorations in the microstructure of Cognition*, The MIT Press, Cambridge, Massachusetts, 1986.

[43] H. P. Nii, Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures, *AI Magazine*, 7(2), pp. 38–53, Summer 1986.

[44] H. P. Nii, Blackboard systems: Blackboard application systems, blackboard systems from a knowledge engineering perspective, *AI Magazine*, 7(3), pp. 82–106, August 1986.

[45] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. Raj Reddy, The Hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainity, *Computing Surveys*, 12, pp. 213–253, 1980.

[46] D. McDermott and C. Sussman, *The CONNIVER reference manual*, MIT Memo 259a, January 1974.

[47] E. M. Riseman and A. R. Hanson, Summary of image understanding research at the University of Massachussets, In *Proceedings DARPA Image Understanding Workshop*, pp. 62–72, Cambridge, Massachussets, April 1988.

[48] C. A. Kohl, Goal-directed control for computer vision, Technical Report 88–22, University of Massachusetts, April 1988.

[49] D. Kuan, H. Shariat, K. Dutta, and P.Ransil, A constraint-based system for interpretation of aerial imagery, In *Proceedings Second International Conference on Computer Vision*, pp. 601–609, Tampa, Florida, December 1988.