USC-SIPI REPORT #178

Hierarchical Representation, Matching, and Search for Some Computer Vision Problems

by

Vadlamannati Venkateswar

May 1991

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
Powell Hall of Engineering
University Park/MC-0272
Los Angeles, CA 90089 U.S.A.

HIERARCHICAL REPRESENTATION, MATCHING, AND SEARCH FOR SOME COMPUTER VISION PROBLEMS

by

Vadlamannati Venkateswar

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Electrical Engineering)

August 1991

Dedication

To my wife, Pratima.

Acknowledgments

I cannot thank my thesis advisor and dissertation committee chairman, Dr. Rama Chellappa, enough. He has been a constant source of excellent guidance, inspiration and support. I would also like to thank the members of my dissertation committee, Dr. Keith Price and Dr. Aristides Requicha, for their support and guidance through my work.

Special thanks to Dr. Allan Weber for assisting with computer system problems, especially in setting up the Texas Instruments Explorer lisp machine used in this work. Mr. Andres Huertas of IRIS provided most of the images used in this work and enthusiastically assisted me with software problems and also with the Nevatia-Babu line finder. Mr. Irwin King of CNE cheerfully helped me during my experiments with KBVision and the Burns line detector.

My deepest appreciation to my fellow graduate students at the Signal and Image Processing Institute, both past and present: especially, Shankar Chatterjee, Tal Simchony, Yi-Tong Zhou, Anand Rangarajan, Gemsun Young, B. S. Manjunath, S. Chandrasekhar, Qinfen Zheng, Tinghu Wu, and Navid Haddadi. They have established a supportive environment for quality research. My thanks also to the staff of SIPI: Erlinda Varilla, Delsa Tan, and Gloria Bullock, for assisting with administrative details.

My appreciation for my sister Indira, brother-in-law Ravi, and little niece Kavita for their support through the Ph.D. program. My most special thanks are reserved for my wife, Pratima, and my parents: Dr. V. Mallikarjuna Rao and V. Varalakshmi. Without their encouragement and support this dissertation would not have been possible.

Contents

List of Figures

Abstract

Representation, matching, and search are central to the task of scene interpretation. Representation refers to how the scene and models are organized for subsequent interpretation tasks, like recognition. A hierarchical representation scheme for scenes and models allows for robust interpretation strategies that exploit the different levels of the hierarchy. In this scheme, features in the scene are composed into more complex objects based on physical and geometric properties and the models are similarly decomposed into simpler features. Such hierarchies are concisely represented using *frames*. For matching, we support a hierarchical strategy where features at the higher level of the hierarchy are matched first and those at the lowest level last. Higher level features are fewer in number and have more structure and so are easier to match. These matches constrain the matches at lower levels. The ambiguities involved in feature grouping and matching result in a search space. As an alternative to relaxation and tree search techniques, we advocate the use of a search scheme based on an Assumption based Truth Maintenance System (ATMS) for exploring this search space. The ATMS assists in simultaneous search in multiple contexts, enforces binary or higher order constraints, assists in symbolic uncertainty reasoning and carries out belief revisions necessitated by incremental additions, deletions and confirmations of feature and match hypotheses.

We design three applications based on these general principles. These include building detection, stereo matching, and motion correspondence. An appropriate representation for these tasks is a hierarchy consisting of lines, vertices, edges, and surfaces. In building detection, a dynamic search scheme based on an ATMS assists in the integration of bottom-up and top-down information in the search for roofs. In stereo matching, a hierarchical matching strategy is used to match the feature hierarchies derived in the two scenes. An ATMS assists in dealing with ambiguities in this matching process. A similar strategy used for motion correspondence results in both point and line matches. This framework can be applied to other applications, like object recognition from 2D or 3D data using 3D models, multi-sensor fusion, etc.

Chapter 1

Introduction

Representation, search, and matching are important issues in computer vision. Representation refers to how the scenes and models are organized for manipulation by subsequent interpretation tasks, like matching and recognition. The scene is usually a 2D array of pixels with intensity (visual images) or depth (range images) values. This pixel information is not amenable for manipulation by interpretation systems. A higher level representation of the scene is desired for subsequent interpretation tasks. Typically such a high level representation is derived from the pixel array in the form of features and their relations. Typical features are regions, curve segments, straight line segments (lines, for short), surface patches, etc. Often it is possible to group these features into more complex features hierarchically. In this scheme lower level features are grouped into higher level features based on geometric or physical properties. Similarly, models can be decomposed hierarchically into simpler features. Frames [1, 2] are well suited for concise representation of hierarchical data and models. The advantage of the hierarchical representation for both scenes and models is that it allows for

robust interpretation strategies that exploit the different levels of the hierarchy. In particular, one can devise a hierarchical matching strategy where features at the highest level of the hierarchy are matched first and those at the lowest level last. Higher level features are fewer in number and have more structure and so are easier to match. These matches will then constrain the matches of their component features at lower levels. This strategy can improve the robustness and efficiency of matching.

The ambiguities involved in the feature grouping and matching process result in a search space that needs to be explored. Several techniques are known for traversing this search space. One technique is to use tree search, where one does a depth-first search of an interpretation tree. The nodes of the interpretation tree correspond to model and data feature pairings. The objective is to find a consistent set of pairings in the space of alternatives. This is done by evaluating unary and binary constraints at the non-leaf nodes of the tree and higher order constraints (like, rigidity) at the leaf nodes. The disadvantage of this method is that backtracking is inherently redundant [3, 4]. A related graph search alternative is to form an association graph where nodes correspond to model and data feature pairings and links connect compatible match pairs. Maximal cliques can be extracted from this graph and evaluated with a goodness measure. The disadvantage of this method is that it is of exponential complexity. One can approximate the maximal clique method with a relaxation based approach, where one checks for local consistency in the association graph in a parallel fashion. This procedure has the advantage of speed, but can get stuck in local minima. Further, certain constraints (like the rigidity constraint) are global and are not well approximated with local consistency measures. We present an

alternative approach to matching in this dissertation, based on an ATMS [5, 4]. The advantages of the ATMS approach is that it allows an opportunistic and efficient traversal of the search space. No backtracking is involved as it performs simultaneous search in multiple contexts. It permits differential diagnosis, where one can compare the results in one point of the search space (a context) with other points directly. This is not possible in other graph search strategies which can only look at one context at a time. The ATMS also assists in the enforcement of binary, ternary or higher-order grouping constraints and assists in the task of belief revisions necessitated by incremental accumulation of evidence during the problem solving process. However, one must keep careful check on the number of contexts, which can in the worst case increase exponentially. So we use a strategy where constraints (binary or higher order) prune the search space and partial solutions are accumulated and *confirmed* thereby restructuring the search space and preventing an unbridled growth in the number of contexts.

We have applied this general framework to three applications:

1. Locating buildings in monocular aerial images.

2. Feature based stereo matching.

3. Motion correspondence of features across several frames of an image sequence.

. In this dissertation, we are interested in images of man-made objects: these include aerial images of urban areas, indoor scenes, factory scenes, etc. These scenes are rich in linear features and so we chose line segments as our basic primitives for all three applications.

For building detection in monocular images, our strategy is to isolate roofs in the picture and use shadow information to estimate their heights. The class of buildings modeled have roofs with orthogonal corners and walls that are perpendicular to the ground plane. Orthogonal and proximate line segments are grouped to form vertices, vertices are grouped into edges and edges into surfaces (edge-rings). Some of the edge-rings are open, so knowledge of the structure of buildings is used to hypothesize edges top-down to close the open edge-rings. An ATMS assists in integrating bottom-up and top-down hypotheses and in enforcing constraints on the shapes of buildings extracted. Vertex to vertex and edge to line shadow correspondences are used to estimate heights of roofs. A wireframe interpretation is constructed for each building hypothesis. The wireframes can be rendered from any viewpoint.

We present a hierarchical feature based stereo matching algorithm in this dissertation. The hierarchy consists of lines, vertices (junctions of line segments), edges (sets of collinear lines with vertex terminations), and surfaces. This is similar to the hierarchy used in building detection, but there is no restriction that vertices must be composed of orthogonal line segments. Matching starts at the highest level of the hierarchy (surfaces) and proceeds to the lowest (lines). Surfaces are fewer in number than other features and they also have more structure. So they are easier to match. Their matches then constrain the matches at lower levels. The hierarchical matching strategy improves the efficiency of line matching and also improves the correctness. To deal with ambiguity, features and matches are treated as hypotheses. At each level of the hierarchy match hypotheses are grouped into islands of certainty based on perceptual and structural relations. These local cliques are then confirmed. An ATMS assists in carrying

out the belief revisions necessitated by the additions, deletions and confirmations of match hypotheses. The ATMS also helps in enforcing binary or higher-order grouping constraints.

We then present an algorithm for motion correspondence over several frames of an image sequence. Feature hierarchies are derived for each frame. The hierarchies from adjacent frames are matched using an algorithm that is similar to the one used for stereo matching. This results in both point and line matches. Match links are maintained over all frames. These matches can be used as input to point and line based motion estimation algorithms.

1.1 Thesis Organization

Chapter 2 reviews previous work and discusses their approach to the problems of matching, representation, and search.

The input to the vision systems designed in this work is in the form of line segments. So the process of line extraction is crucial. We have developed a line extraction scheme that produces high quality line descriptions suitable for use in these systems. Chapter 3 describes this line extraction algorithm. Results and comparisons with other popular line extractors are also presented in this chapter.

Chapter 4 deals with the problem of detecting buildings in monocular aerial images. The problem of detecting buildings is posed as a graph search problem. The nodes in this graph are not static; new nodes can be added top-down and old ones deleted. A dynamic search strategy is described that can deal with incremental additions and deletions of nodes. This strategy combines both bottom-up information and top-down knowledge of structure of buildings. We also describe

how shadow evidence is used to estimate the heights of buildings. Real image examples are given.

Chapter 5 develops a stereo matching algorithm that matches line segments in stereo pair. We review existing methods and their shortcomings. A hierarchical matching strategy based on the formation of local cliques is presented. This strategy improves the robustness and efficiency of matching. We also motivate the need for truth maintenance to deal with the uncertainty in the system as well as in enforcing binary, ternary or higher order constraints. We then present results of matching several stereo pairs. Chapter 5 also describes a method for feature based motion correspondence over several frames. This algorithm tracks lines and points over several frames and can be used as an input to feature based motion estimation systems. Results are presented for an image sequence.

Conclusions and directions for future research are presented in Chapter 6.

1.2 Thesis Contributions

The main contributions of this dissertation are as follows:

1. A new line extraction algorithm that produces high quality line descriptions from aerial images is developed [6].

2. We demonstrate the use of an ATMS in integrating bottom-up and top-down strategies when searching for buildings in aerial images.

3. We develop a hierarchical matching strategy to improve both the robustness and efficiency of matching in stereo and motion correspondence.

4. We describe a technique for obtaining both point and line correspondences over several frames of an image sequence.

5. We present an ATMS approach to dealing with the uncertainties involved in feature grouping and matching and as an elegant means of enforcing constraints.

Chapter 2

Background

In this chapter we delve deeper into the issues of representation, matching, and search and review how current vision systems have addressed these issues.

2.1 Representation

By representation we mean both scene and model representation. The scene and the models can be either 2D or 3D. The traditional approach to scene representation is to isolate primitives from the image and establish relations between them. The primitives together with their relationships form a relational graph or semantic network. The primitives form the nodes in the graph and their relations form the links. This graph is then used in subsequent processes like matching, recognition, etc. The models are also represented similarly as relational graphs and the process of finding a correspondence between the models and the image is treated as a graph matching problem. This is the approach taken by Nevatia and Price [7] in their system for locating structures in aerial images. They represent

the scene and model with line segments and regions as the primitives, and neighbors, part-of, above, etc. as their relations. Frequently, it is possible to group low level primitives into more complex structures based on their *structural* relationships. The advantage of such a grouping is that it simplifies subsequent interpretation tasks, like recognition and matching. Such a multi-level representation scheme was used in VISIONS [8]. The image is segmented into a Region, Segment, Vertex (RSV) graph. Image models are represented with schema networks. Schemas are further decomposed hierarchically into objects, volumes, surfaces, regions, segments, and vertices. The hierarchical representation of images and models allows for a complex recognition strategy that uses knowledge sources for interaction between different levels of the hierarchy. This strategy of hierarchical problem decomposition reduces system complexity, permits reusability of parts, and increases system robustness by permitting accumulation of intermediate evidence [9]. Andress and Kak [10] use a hierarchy made up of vertices, edges, faces, and objects. Dempster-Shafer theory is used for evidential reasoning during the grouping process. This system is limited to the task of labelling edges from an image model. In the 3D-MOSAIC system [11], a 3D wire-frame interpretation consisting of a surface-edge-vertex structure graph is derived from monocular images or stereo pairs using task specific assumptions. When matching a stereo pair only vertices are matched and the results are used to derive 3D wireframes using knowledge of urban scenes and structures of buildings. The 3D FORM [12] was designed as an extension to the 3D MOSAIC. It stores explicit models for objects using a hierarchical representation based on sub-part relationships. Frames are used to model the objects in a hierarchical fashion. ACRONYM [13] also models objects as compositions of generalized cylinders using frames to represent part,

sub-part relationships. ACRONYM is a general vision system for detecting 3D objects from their 2D projections. The scene, however, is not represented hierarchically. It is represented as an observation graph consisting of ellipses and rectangles and their relationships. SIGMA [14] also uses frames to represent the models hierarchically. This system is limited to the task of identifying simple buildings in aerial suburban scenes. Recognition is achieved by composing primitive hypotheses like rectangular blobs, etc. into more complex hypotheses using domain specific information stored in frames. Lim and Binford [15, 16] derive a hierarchy of features starting with edge pixel primitives and using structural information. Edge pixels are grouped into curves, curves are intersected to form junctions, adjoining curves are pieced together to form surfaces and surfaces are grouped into objects. The hierarchies are obtained separately from the right and left images of a stereo pair and then matched.

Sometimes, primitives can be composed into a hierarchy based on *perceptual* relations rather than structural. Perceptual relations are those relations that are viewpoint invariant. These include parallelism, collinearity, and proximity. Lowe [17] groups perceptually related line segments. Compound structures are identified by searching the graph of primitive relations for specific combinations of relations that share line segments. These include, trapezoids, parallelograms, etc. The models are specified as wireframes using 3D line segments. Compound structures are also identified in the models. Groupings in the image are used to index into the models. The viewpoint is solved and the model projected onto the image for verification. Mohan and Nevatia [18] use a constraint satisfaction network to derive perceptually significant structures from line segment primitives.

Line segments are grouped hierarchically into parallels, u-contours, and rectangles. Rectangles derived from two scenes are matched to give 3D descriptions.

The hierarchies need not always be based on perceptual or structural relations. One can also use domain dependent knowledge to group primitives hierarchically based on *functional* relations. When this is the case, recognition occurs as an automatic byproduct of the hierarchical grouping process. McKeown et al. [19] use a hierarchy made up of region fragments, functional areas, and models in their aerial image interpretation system. Functional relations are used in the grouping. For example; runway and tarmac regions are combined to form a runway functional area. OPS5 [20] working memory elements are used as the language for representing the elements in the hierarchy and their relations. Nagao and Matsuyama [21] use a blackboard system to interpret multi-spectral high-altitude aerial images. Elementary regions are first extracted from the image based on multi-spectral properties and these are grouped into characteristic regions. Object detection subsystems then combine these characteristic regions into objects.

2.1.1 Frames

It can be seen from the above discussion that frames are a popular form of representation for hierarchical data [12, 13, 14]. To understand why, we need to take a detailed look at frames. Minsky [1] originally proposed frames as a way of describing prototypical situations with instances inheriting the defaults of the prototype. But his description was more at an abstract level and less at a computational level. The popularity of frames increased with the computational models

for frames developed in several frame languages, including KRL [22] and KEE [2]. Frames are modeled as data structures with slots. These slots can be filled with values. Slots can be *relation* slots or *attribute* slots. Attribute slots store the properties of the frame and relation slots relate the frame with other frames. Each relation has an inverse slot associated with it. For example; *instance-of* is the inverse of *has-instance*. Relation slots effectively set up bidirectional links between frames. The result is a semantic network. However, frames have more features. These includes the features of *inheritance* and *procedural attachment* which bring them closer to Minsky's ideas. Frames can be arranged into taxonomies with specific frames inheriting the slot values of generic frames. This permits description of prototypical situations or objects. Procedures can be attached to frames in the form of *methods* and *active values*. Methods are invoked by passing *messages* to frames. Active values are *demons* that are triggered automatically by slot operations like additions or deletions of slot values. Procedures are also inherited down the frame hierarchy. There are definite advantages with this kind of a data structure. *Specialization* through *inheritance* reduces the need to specify redundant information. The message sending paradigm supports an important principle in programming: *data abstraction* [23]. Message passing together with specialization creates an object-oriented framework that is ideally suited for modifying and extending programs [23].

2.2 Search

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 to lead to a solution. It is the task of the problem solver to find the correct solution in this search space, intelligently, and efficiently. One way to deal with alternatives is to choose one and eliminate the rest by a local analysis. However, this could lead to incorrect solutions. Consider the simple example in Figure 2.1. The task is to compose

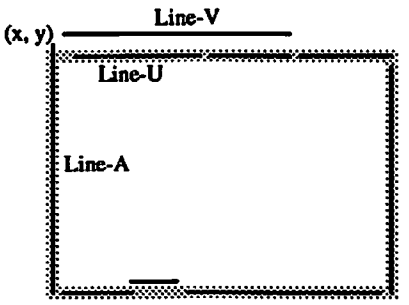


Figure 2.1: Composing rectangles from line segments.

rectangles from the line segments in the figure. Some of the line segments are fragmented. One way to compose rectangles is to first identify potential corners and then group them. Just based on the local situation near 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 approach based on such a local heuristic will fail. Situations more complex than that in Figure 2.1 are frequently encountered in vision. Local context is not sufficient for disambiguation. The reasons are manifold. One is because of accidental juxtaposition of events. In Figure 2.1, 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 line segments and is very similar. Another problem 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 example: 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 higher level modules.

The simplest search strategy is to follow each path independently and see if any of them leads to a solution [4]. Depth-first search can be used to improve the efficiency of this search. In this approach the space of all possibilities is treated as a tree. We start at the root of the tree and any alternatives we encounter are treated as branches of the tree that lead to new nodes. The new nodes may then further branch into more nodes. The idea is to explore this tree systematically. Depth-first search starts at the root of the tree and explores along the first branch encountered at each node. If the path leads to a dead end or a contradiction then the search backtracks to the last node at which it branched. Then the next branch from this node is explored. For this reason depth-first search is also called chronological backtracking, as backtracking is done to the node that was encountered chronologically just before the current node. This kind of backtracking is known to have problems [3, 4]. In particular when the system backtracks because it encounters some contradictions, it forgets any inferences made in that portion of the search space. These inferences will

be rediscovered at other places. Also the underlying reasons for contradictions are not stored so that these contradictions may be encountered again and again. As a solution to the "forgetting" problem, Stallman and Sussman [24] developed dependency-directed backtracking. In this scheme, dependency records for each inference are maintained. These records link each inference to its antecedents. When a contradiction is encountered, the dependency records are consulted to backtrack to the most recent selection which actually contributed to the contradiction, instead of backtracking in chronological order. This prevents futile backtracking. Further, these dependency records are bidirectional and are used to reinstate previously derived information in different portions of the search space. This avoids rediscovery of inferences. Also when a contradiction is encountered the underlying assumptions that actually led to the conflict are stored for future reference as *nogoods*. These *nogoods* are used to prevent rediscovery of contradictions. Dependency-directed backtracking was exploited by Doyle [25] in his justification based Truth Maintenance System (JTMS). A JTMS works with nodes. Each node corresponds to a problem solver datum. Associated with each node is a justification which summarizes how the node originated. Truth maintenance is a procedure that decides which problem solver datum is to be believed (in) and which disbelieved (out). It uses dependency directed backtracking for this process.

de Kleer [4, 26] proposed an ATMS as an improvement over a JTMS. The ATMS works by manipulating assumption sets. Assumptions are primitive data from which all other data are derived. The ATMS works on the principle that they can be manipulated far more conveniently than the datum sets they represent. The ATMS simplifies truth maintenance and eliminates backtracking.

When using the ATMS, all contexts (points in the search space) are simultaneously visible to the problem solver. There is no single-state restriction. This permits differential diagnosis, where one solution can be directly compared with other solutions at different points in the search space and the "best" interpretation chosen. This is not possible in any scheme that uses backtracking to move from one point of the search space to another (including JTMSs). The ATMS will find all the solutions that a pure enumeration technique would. Further, it improves the efficiency of the problem solver without sacrificing coherence and exhaustivity. In our application we use ART [5] which has an ATMS that is similar in functionality to de Kleer's system.

Truth Maintenance Systems (TMSs) are symbolic approaches to uncertainty management. Both numerical and symbolic approaches are known in the literature to deal with uncertainty. For a review, see [27]. Popular numeric approaches include the Bayes net approach [28], Dempster-Shafer theory [29], and fuzzy logic methods [30]. Symbolic approaches include TMSs [25, 4, 31]. In TMSs, uncertainty is dealt with by making *assumptions* (*hypotheses*) whose beliefs can later be revised as new evidence accumulates. Unlike numerical methods, symbolic methods cannot represent degrees of conflict (but see [32], which integrates Dempster-Shafer formalism into a TMS), but they permit explicit reasoning about the assumptions that actually lead to the conflict (conflict resolution). If rigid constraints can be formulated (as in the case of stereo matching, motion correspondence, and object recognition) then there is more of a need for an approach that can reason about the conflicts produced by these constraints than one that can deal with degrees of conflict. Then a TMS is more appropriate to deal with uncertainty. Another advantage in using a TMS is that it provides

mechanisms for the automatic construction of the inference network. Numerical approaches [28, 33, 34] assume that an inference network is somehow constructed and already in place.

The 3D MOSAIC [11] system resembles a JTMS in the sense that dependencies are implicitly stored in the form of constraint links in a surface-edge-vertex graph. However, when the links are deleted so are the dependencies, so rediscoveries of inferences and contradictions may not be avoidable and dependency-directed backtracking is not possible. So virtually all the power of a JTMS is lost. Provan [35] used an ATMS to solve the problem of detecting puppets in a set of rectangles. Bowen and Mayhew [36] used the ATMS to build consistent wireframes from fragmented 3D line segments and circular arcs. A stereo system first matches edge points in a stereo pair and generates 3D coordinates for these points. Then these 3D points are grouped into line segments and circular arcs. However, these descriptions are fragmented because of the nature of edge detection; for example, at corners. Rules are used to compose these fragments into a Region, Edge, Vertex graph (REVgraph). A TMS is used to explore the alternatives during the grouping process. Results are shown for simple wireframes. The problem of search occurs most often in vision during matching. We review this problem in detail next.

2.3 Matching

The issue of matching with constraints occurs often in vision: 2D image to 3D model matching [13, 17], 3D data to 3D model matching [37, 38, 39, 40, 41, 39, 42], image to map matching [7], 2D shape matching [43], and image to image

feature based matching [44, 45, 46, 47, 48, 49, 50]. Matching in these systems is posed as the problem of matching graph descriptions while enforcing constraints. In the sequel we refer to one graph description as the object graph and the other a label graph. The objective is to find an assignment of a label to each object. Unary constraints based on feature similarities are first used to generate an initial set of matches. No global context is used in this stage and this results in several spurious matches. Then binary or higher order constraints (based on the compatibility of matches) are used while sifting through the space of competing and supporting matches. Constraint satisfaction techniques like backtracking [51, 52], discrete relaxation [3, 44, 53, 54] or continuous relaxation [18, 50, 55] can be used to sift through the space of competing and supporting matches. Grimson [51] suggests an approach based on backtracking over an interpretation tree while enforcing binary constraints. The nodes of the interpretation tree correspond to model and data feature pairings. The aim is to detect 3D objects from 3D data or 2D objects from 2D data. However, as already mentioned backtracking is inherently redundant. Modifications have been suggested to improve backtracking. These include branch and bound techniques [56, 51], where the current best interpretation is maintained and any path worse than that is immediately cut off. Tree search with backtracking is also used in the object recognition systems of Faugeras and Hebert [40], Oshima and Shirai [41], and Fan, Medioni, and Nevatia [42]. We now focus on the most recent of these [42]. The objective in this system is to match 3D objects to 3D data. Both are described in terms of surface patches and their relations. A measure of goodness is associated with each compatible model and scene node pair. Depth first search is used to build a set with 4 pairs. The best set is identified and expanded again in depth-first

fashion till a "good" solution is obtained. Tree search with backtracking is also used in the 3D-FORM [12]. First, low level primitives are grouped into the most general class of objects (3D-OBJECT). However, the issue of alternatives and errors during the grouping process is not addressed. The 3D-OBJECTs are then specialized by searching down the IS-A and PART-OF hierarchies using a tree search with backtracking. Simple examples involving hand generated 3D wireframes are presented.

Another approach to deriving a subset of matches from all the candidate matches is the maximal clique method [57]. Here an association graph is constructed from all the matches. Nodes in the graph correspond to candidate matches. Compatible matches are connected by links. Maximal cliques in the graph correspond to solutions of the matching problem. These maximal cliques are extracted and one of them chosen as the "best" based on some measure. This approach was used in the local-feature-focus method of Bolles and Cain [58], the 3DPO part orientation system [39] and the stereo matching algorithm of Horaud and Skordas [47]. However, the problem of listing all maximal cliques in the graph is of exponential complexity. Approximations to these clique finding algorithms are possible, mainly through relaxation algorithms. In these algorithms local functions are used to iteratively approximate some global function. Two forms of relaxation algorithms exist: discrete relaxation and continuous relaxation. Medioni and Nevatia [44] use discrete relaxation to match line segments in two scenes. An initial set of line matches are generated based on similarity of orientation and proximity. At each iteration only those matches with a sufficient number of compatible matches in the neighborhood are retained. Compatibility is measured by similarity of relative position of the line segments and their

matches in the two images. Discrete relaxation is also used in ACRONYM [13], a general vision system for detecting 3D objects from their 2D projections. Objects are modeled as compositions of generalized cylinders, using frames to represent sub-part hierarchy. An observation graph consisting of ellipses and rectangles with their relationships is generated bottom-up from a 2D image. A geometric reasoning system generates a prediction graph from the models. Maximal subgraphs of the observation graph are matched to subgraphs of interpretation graph using a variation of Waltz's [53] discrete relaxation algorithm. Herman and Kanade [11] also merge wireframes derived from separate scenes using a discrete relaxation algorithm. First connected groups of edges and vertices in the wireframe are identified as wire frame objects. These wireframe objects are matched by looking for matches of their constituent edges and vertices. Another form of relaxation used in matching is continuous relaxation. Price reviewed continuous relaxation matching techniques in [50]. These techniques differ from the discrete case in that they can deal with measures of likelihoods of matches and not just zero or complete likelihood. Likelihoods are computed based on feature similarities. A local function measures the compatibility of each match with its neighbors. At each iteration, likelihoods are altered to increase this local compatibility measure. Matches with high possibilities are chosen as a solution. This technique was used in matching images to models and also images to images [50, 59].

An alternative to backtracking over a tree is a lattice search, where several paths are explored simultaneously. Backtracking is completely avoided in the beam search used in 3D MOSAIC [11] to match junctions in a stereo pair. This is a sub-optimal search strategy. The system follows the N best partial paths

(where N is a fraction of the total possibilities) simultaneously. The junctions in the left image are arranged in a file with neighboring vertices adjacent to each other. Then candidate matches of each junction are identified and listed in a column along with the junction. Any path in this network that visits one candidate in each column gives a unique set of matches. Costs are associated with each node and also the paths between adjacent nodes. Beam search starts with column 1 and retains only the N best partial paths at each column. This search strategy may not find the lowest cost path, as this path may be discarded at any column where it is not among the N best. In SIGMA [14], lattice search is used to combine related primitive hypotheses into situation sets forming a situation lattice. Conflicting interpretations coexist in the situation lattice and no conflict resolution rules are defined. Lattice search is also used in ATMSs where all contexts are explored simultaneously. Bodington, Sullivan, and Baker [60] experimented on the use of an ATMS for matching a 3D model to a 2D image. An interpretation tree approach [37] based on backtracking is shown to be less desirable than an ATMS for this application. Heuristics are described to reduce the extra memory cost incurred in using an ATMS. We used an ATMS in three applications. These include building detection [61], stereo matching [62], and motion correspondence [63]. The advantage in using a TMS for the image matching problem is that binary, ternary or higher order constraints are easily incorporated. Also, the problem solver has more control of the path to a solution and there is a possibility of avoiding local minima. Further, no backtracking is involved when using the ATMS.

2.4 Applications

We consider three applications in this dissertation that implement our ideas on representation, matching, and search. These include: location of buildings in aerial images, feature based stereo matching, and feature based motion correspondence.

Building Detection:

Detection of buildings in aerial images is an important task in photo interpretation of aerial images. Several systems have been developed for this problem. In the 3D MOSAIC system [11] buildings are modeled as block shaped objects. The system can reconstruct urban scenes from both monocular images and stereo pairs. Line segments are used as the basic primitives during processing. In monocular analysis the vertical vanishing point is assumed to be known and all lines passing through this point (when extended) are identified as vertical lines. This information is used to predict the heights of structures in the image. However, vertical lines are heavily foreshortened in aerial images and are not reliably detected and so are not reliable cues for height estimation. Huertas and Nevatia [64] use shadows to estimate heights of buildings. Buildings are modeled as unions of rectangular parallelopipeds. Roofs of buildings are detected by forming chains of compatible corners. Heuristics are used to close open chains of corners. The system makes unilateral decisions at several points in the analysis based on local context. A more preferable approach would be to explore all alternatives and use global context later in the search process to resolve ambiguities. For example, when finding corners, a local spiral search is used at end points of line

segments to choose the nearest segment. An approach that explores the possibility of corners with other line segments in the vicinity would be more robust. Later the corner that has a better global context (for example, a corner that is part of a closed chain of corners with shadow evidence) can be confirmed and the others deleted. BABE (Builtup Area Building Extraction) [65] also uses shadow evidence but does not assume that the sun angle or shadow intensity are known. The system models buildings as boxes. Liow and Pavlidis [66] also use shadows. They integrate region growing and edge detection to detect complex roof shapes. Shadows are used in verification, but not in reconstruction. Hwang et al. [14] propose a general framework based on the integration of hypotheses for building vision systems. They also describe SIGMA, a specific implementation using this framework for detecting simple buildings in 2D suburban scenes. The problem of reconstructing the shape of the buildings is not addressed.

Feature based Stereo Matching:

Stereo matching is the process of fusing two images taken from different viewpoints to recover depth information in the scene. The process involves identifying corresponding points or regions in two views and using their relative displacements together with camera geometry to estimate their depth. Two approaches are known in stereo literature for determining this correspondence: area based and feature based. Area based methods find corresponding local regions in the images by computing cross correlation [67]. They suffer from the limitation that accurate determination of depth is not possible, because of averaging over neighborhoods. Feature based methods involve determining certain features in the images and matching these features. Depth can be accurately determined at these features. Interpolation techniques are then used to determine the depth at

other points. In this dissertation we are primarily concerned with feature based methods. Different researchers have used different features as the matching primitives. Marr, Poggio, and Grimson [68, 69] match zero crossings of Laplacian of Gaussian operators. Uniqueness and continuity constraints are used to choose between competing matches. Mayhew and Frisby [70] match zero crossings and peaks of convolution profiles. A *figural continuity* constraint is used to favor those matches that yield similar disparities for connected features. Baker and Binford [71] use edges for matching, with the constraint that the ordering of edges in the two scenes be preserved. Edge connectivity is used to enforce continuity of disparity along edge contours. Ohta and Kanade [72] use edge-delimited intervals of scanlines as elements to be matched. These intra-scanline matches together with inter-scanline edge connections are used in a dynamic programming framework to derive matches. Hoff and Ahuja [73] integrate matching and interpolation, which traditionally were treated as separate problems. They match edges and use a surface smoothness constraint to select subsets of matches. Hsieh, Perlant, and McKeown [74] integrate the disparity results obtained by a feature based method with those obtained by an area based method. Peaks and valleys of intensity profiles of epipolar lines are matched in the feature based method. Herman and Kanade [11] use L-junctions (cotermination of line segments) as features for matching. Domain-based constraints are used to restrict the number of matches. The space of potential matches is then searched using a beam search to find a mapping. Mohan and Nevatia [18] use rectangle features as primitives for matching.

Recently, line segments have gained popularity as features to be matched across scenes. Line segments are collections of edge pixels that are connected in

a straight line in the scene. The advantage in matching line segments instead of edge pixels is that matching complexity is reduced, because collections of edge pixels are matched instead of individual pixels and the continuity constraint is automatically enforced across straight edge contours, because they are treated as a unit when matching. The epipolar constraint [75] is used to restrict the number of line matches. Lines that can be matched must span the same set of epipolar lines (actually this is rather restrictive as lines extracted from an image are frequently fragmented, so a small overlap of the epipolar extents is usually considered sufficient for matching). However, much ambiguity remains because a line in one scene can match several lines in the other scene, in spite of the epipolar constraint. So there is a need to consider a more global context to disambiguate line matches. Medioni and Nevatia [45] use a minimum-differential-disparity criterion for disambiguation. Line matches that are adjacent and have similar disparities support each other in a relaxation framework. Ayache and Faverjon [46] cluster adjacent matches with similar disparities. A prediction and propagation algorithm is used to generate these clusters by first deriving an initial set of strong matches and then looking in their neighborhoods for similar-disparity matches. Large clusters correspond to a solution. Both [45] and [46] are based on the idea that adjacent line segments should have similar disparities. This is, however, a weak constraint because adjacent lines in the scene need not have similar depths. This constraint favors frontoparallel surfaces [73]. Horaud and Skordas [47] reduce matching ambiguity by also considering feature relations between line segments (collinear-with, same-junction-as, left-of, etc.) in the matching process. They use the maximal clique method discussed earlier.

As discussed earlier, this procedure is of exponential complexity. Lim and Binford [15, 16] only match bodies after deriving a hierarchy consisting of edgels, curves, junctions, surfaces and bodies. The result is that only those features that can be grouped into bodies can be matched. Also, no ambiguity is assumed in the matching procedure described in [16]. Chung and Nevatia [76] also match a hierarchy consisting of edges, curves, ribbons and junctions. They also present techniques to deal explicitly with occlusion effects. A constraint satisfaction network [18] is used to deal with the ambiguities in the matching process. However, relaxation is only an approximation technique and there is a danger of getting stuck in local minima.

Feature based Motion Correspondence:

Two types of features are popularly used in feature based motion estimation: point features and line features. Techniques are known that can estimate motion based on point correspondences over two [77, 78, 79, 80], three [77] or several frames [81, 82]. Line based techniques need correspondences over three [80, 83, 84, 85, 86] or more [48, 49] frames to estimate motion. Techniques that use both point and line correspondences are also being developed [87].

Several methods have been proposed for obtaining point correspondences. Barnard and Thompson [88] developed a relaxation based matching for matching point patterns in two scenes. An initial network of possible matches is constructed and a disparity is associated with each match. A subset of these matches are selected by a relaxation algorithm that prefers smoothness of disparity in the scene. However this is a weak constraint, because interest points in the scene are usually not contiguous but are in fact disconnected and sparsely distributed. Such points need not have similar disparity. Ranade and Rosenfeld [89] also

developed a relaxation based algorithm similar to [88]. Fang and Huang [90] extended this matching technique to account for scale changes between images. Sethi and Jain [91] derived trajectories of points over an image sequence by using smoothness assumptions. Chandrasekhar and Chellappa [92] interleave matching, estimation, and prediction of points across an image sequence. The extraction and matching of feature points are done using a Gabor wavelet representation and an extended Kalman filter is used for estimation.

Techniques are also available for generating line matches. Medioni and Nevatia [44] match linear features in a discrete relaxation framework. Compatibility is measured by similarity of local structure in the two images. Deriche and Faugeras [49] combined matching and motion estimation for line features. A Kalman filter was used to recursively update motion parameters and these parameters and their covariances were used to define a search area for matching in the next scene. Crowley, Stelmaszyk, and Discours [48] used a similar approach. An image plane flow model was maintained. Lines were expressed as parameter vectors and these parameters were updated using a kalman filter. Confidence factors were associated with line segments to control their existence in the flow model.

Chapter 3

Extraction of Straight Lines

3.1 Introduction

The amount of data contained in any image is large. The purpose of the low level module is to reduce this data by abstracting a small number of primitives. These primitives are then fed to higher level modules. Typical primitives are regions, curve segments, ellipses, lines, etc. The kinds of primitives to be extracted depend on the nature of the interpretation task. For example, consider the system developed by Nagao et al. [21] for the interpretation of high altitude multi-spectral images. The system classifies a multi-spectral image into different object regions (crop fields, roads, buildings, etc). Consequently, the input primitives for the higher level module are *elementary* regions extracted from the image using spectral properties. In [19], 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). So both lines and regions are used as low-level primitives. In ACRONYM [13], a general vision system which has been

tested on aerial images of airport scenes, objects are modeled as a restricted class of generalized cones. The swept surfaces of generalized cones generate ribbons in the image and end surfaces generate ellipses. Therefore the low level primitives are ribbons and ellipses extracted from the image. In [11, 18, 64, 61], the task is to detect buildings in aerial images that are bound by planar surfaces. The input primitives are lines extracted from the image. Those lines corresponding to object edges are identified and grouped together to generate wireframe descriptions of buildings.

In this dissertation, we are primarily concerned with the task of interpreting images of man-made objects (buildings, airports, roads, offices, etc). Most of the information on the structures of man-made objects is in the boundaries. So line segments are the ideal primitives for these domain, since one can detect contours of man-made objects as piecewise linear segments. Several methods are known in the literature for the detection of lines and they are briefly reviewed below.

Hough Transform. In the Hough transform method [93], each edge pixel is indexed into a curve in a parameter space (ρ-θ space; ρ is the perpendicular distance of the line from the origin and θ is the slope of its normal), where each point on the curve specifies a straight line passing through the edge pixel. Collinear edge pixels map into concurrent curves. The parameter space is quantized and clusters in the parameter space then correspond to straight lines. An improved version is described in [94], where the edge pixel direction is used to reduce the mapping of each edge pixel to a single point in the parameter space. The Hough transform method has the drawback that it fits straight lines to collinear points without any regard to their

spatial contiguity. Furthermore, the likelihood of large clusters occurring at random is quite high. The noise sensitivity of the Hough transform has been analyzed in [95] and more recently in [96]. The Hough transform is a general method and it can be used to detect arbitrary shapes and not just straight lines [97]. A comprehensive survey of the Hough Transform literature can be found in [98]. Subsequent methods developed specifically for straight line extraction were found to be qualitatively better. These methods are described below.

Nevatia-Babu method. In the Nevatia-Babu method [99], the image is first convolved with six 5×5 gradient masks. The output is then thresholded and thinned to produce edge pixels. These edge pixels 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 an iterative end point fitting method.

Burns method. In Burns et al. [100], 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.

When typical outputs of these line detectors are observed for images of man-made objects we identify a few problems.

1. A significant percentage of the lines do not correspond to object boundaries. This increases the search space for object detection leading to wasted time in the higher level modules.

2. Many straight object boundaries are fragmented into small lines. This further complicates the task of object detection.

Thus there is a need for algorithms that can improve on these two points. In this chapter, we describe a simple and efficient algorithm that attempts to do just that. This line extraction algorithm has its roots in a preliminary version presented in [101]. It has been successfully used as the front end of several systems [61, 62, 63].

The line extraction algorithm has five stages.

1. The first stage is edge detection. The edge pixels in the graph are identified and the contrast as well as edge direction at each edge pixel specified. We use the Canny edge detector [102] for this stage.

2. The second stage is a connected components algorithm. The edge pixels are the nodes in the graph. Edge pixels that are proximate and have similar edge directions are connected. The objective is to identify the connected components (lines) in this graph. This is accomplished by a labelling algorithm that assigns line labels to each node and thereby generates a label image. Nodes (edge pixels) that belong to the same connected component (line) are assigned the same line label. At the same time the algorithm

incrementally builds an internal description of the lines in the form of a record with fields for the end points, average contrast and the pixel-support of the line. This algorithm uses a small neighborhood (3×5) of the edge and label images at any time and is therefore efficient. Section 3.2 illustrates the labelling algorithm.

3. The edge detector that is used in the first stage does not work well at the ends of lines where they meet with other lines to form corners. The reason is that the edge detector is optimized to detect step edges, but the image is not a perfect step at corners. As a result the second stage often fails to link the edge pixels towards the ends of lines. Section 3.3 describes the third stage of line extraction that attempts to do some remedial work at the corners.

4. Because the labelling algorithm works with a small neighborhood, a few object contours (especially those interrupted by noise or poor contrast) are fragmented into smaller lines. So the fourth stage links these fragmented and collinear lines together by using the label image as a spatial index to quickly search for collinear lines. This stage is described in Section 3.4.

5. The objective of this line extractor is to identify the straight line contours of man-made objects. Ideally we would like to extract these contours exactly and nothing else. An examination of the results produced at the end of previous stages shows that there are many spurious lines that have no physical significance. Section 3.5 describes the final stage of the algorithm that attempts to eliminate most such lines. Section 3.6 gives some real image examples and comparisons with the methods presented in [99, 100].

3.2 A Scan and Label Algorithm

The input to the system is in the form of an edge image produced by an edge detector. The edge detector must identify edge pixels as well as specify the contrast and edge direction at these pixels. We use the Canny edge detector [102] to generate the edge image. One advantage of using the Canny edge detector is that the hysteresis thresholding that is used as part of edge detection results in an edge image that has fewer breaks in continuity along edge contours compared to conventional thresholding. We then quantize the edge direction of each edge

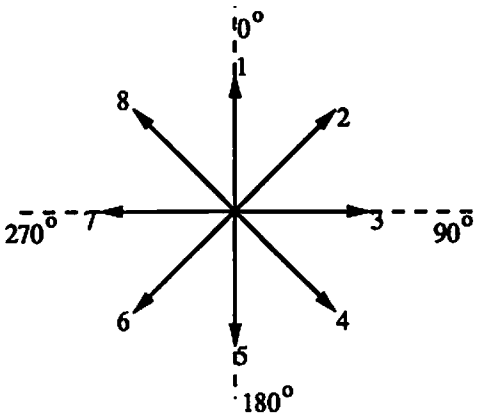


Figure 3.1: Quantizing the edge pixel directions.

pixel into one of eight directions, as shown in Figure 3.1. The reason for the quantization is that the connected components (scan-and-label) algorithm to be described below uses templates based on quantized edge directions to link edge pixels.

We want to link the edge pixels based on similarity of edge-direction. This is essentially a connected-components (of a graph) algorithm where the edge pixels are the nodes and the connections are based on similarity of edge direction. There are two issues to be considered:

1. How to devise an efficient connected-components algorithm:

   The complexity of searching for connected components in a graph is a linear function of the number of nodes and the number of links [103]. We designed an efficient scan-and-label algorithm that takes advantage of the embedding of the graph in a two-dimensional image plane. The coordinates at each node (edge pixel) in the graph index it into a 2-D image plane. The complexity of this algorithm is a linear function of its nodes only (links excluded). Further, this algorithm generates a straight line description of each connected component. The algorithm is explained later in this section.

2. How to specify the connections between the nodes of the graph:

   Intuitively two pixels are connected if they have the same edge direction and the line joining them also has the same direction. We observed the patterns formed by the edge pixels of straight line contours over several images and noticed a general trend among them. This a priori knowledge was then encoded as a set of heuristic templates based on quantized edge directions. These pre-defined templates implicitly specify the connections between the nodes (edge pixels) in the graph. Details of these templates follow later in this section.

The scan-and-label algorithm scans the edge image LRTB and assigns a line label to each edge pixel. Simultaneously it builds an internal description for each line label. This description is a record that has fields for the start and end points, the average contrast along the line and the pixel-support of the line (number of pixels assigned this line label). The idea of this scan-and-label process is to assign the same line label to edge pixels that fall along a straight line contour.

1. How to devise an efficient connected-components algorithm:

   The complexity of searching for connected components in a graph is a linear function of the number of nodes and the number of links [103]. We designed an efficient scan-and-label algorithm that takes advantage of the embedding of the graph in a two-dimensional image plane. The coordinates at each node (edge pixel) in the graph index it into a 2-D image plane. The complexity of this algorithm is a linear function of its nodes only (links excluded). Further, this algorithm generates a straight line description of each connected component. The algorithm is explained later in this section.

2. How to specify the connections between the nodes of the graph:

   Intuitively two pixels are connected if they have the same edge direction and the line joining them also has the same direction. We observed the patterns formed by the edge pixels of straight line contours over several images and noticed a general trend among them. This a priori knowledge was then encoded as a set of heuristic templates based on quantized edge directions. These pre-defined templates implicitly specify the connections between the nodes (edge pixels) in the graph. Details of these templates follow later in this section.

The scan-and-label algorithm scans the edge image LRTB and assigns a line label to each edge pixel. Simultaneously it builds an internal description for each line label. This description is a record that has fields for the start and end points, the average contrast along the line and the pixel-support of the line (number of pixels assigned this line label). The idea of this scan-and-label process is to assign the same line label to edge pixels that fall along a straight line contour.

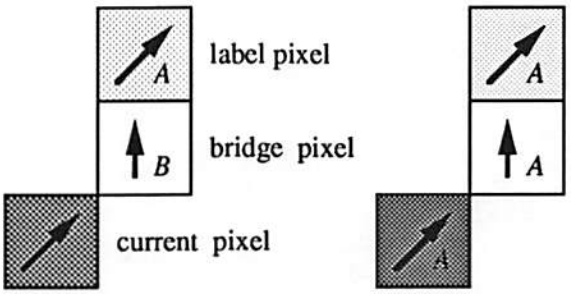
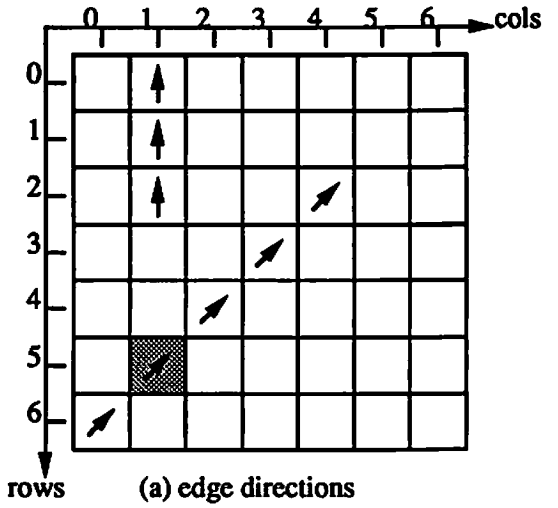


Figure 3.2: (a) Pixel types. (b) Result after labelling the current pixel. Line labels are shown in the lower right corner of the pixel boxes. Arrows represent edge directions.

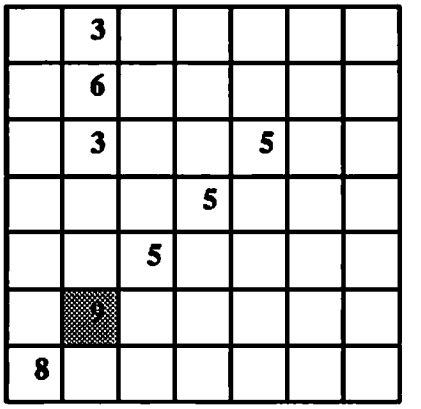
So when an edge pixel is visited, the algorithm checks if it inherits the line label of a neighboring edge pixel based on continuity in edge direction. Consider Figure 3.2(a). Three kinds of edge pixels that participate in this continuity test can be defined. The *current pixel* is the edge pixel that is to be labelled. The *label pixel* is the edge pixel whose line label the current pixel can inherit. The *bridge pixel* is the edge pixel that bridges any gap in spatial continuity between the label pixel and the current pixel. At the time that the current pixel is to be assigned a line label, the label pixel and the bridge pixel would have already been assigned line labels by the scan-and-label algorithm (The labels A and B in Figure 3.2(a)). The algorithm checks the following in sequence:

1. The current pixel must have the same direction as the label pixel. This constraint ensures continuity of direction.

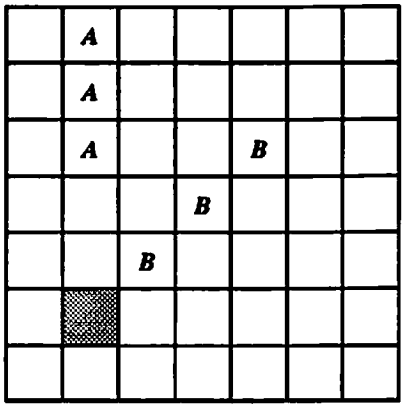
2. The label pixel must be the end point of the line that it is a part of. This constraint ensures that a line can be incrementally extended only from its end points.



(a) edge directions



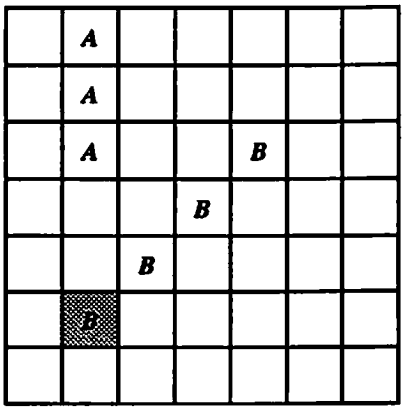
(b) edge magnitudes



A → start = (0, 1)
end = (2, 1)
pixel-support = 3
avg-contrast = 4

B → 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



A → start = (0, 1)
end = (2, 1)
pixel-support = 3
avg-contrast = 4

B → start = (2, 4)
end = (5, 1)
pixel-support = 4
avg-contrast = 6

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

Figure 3.3: The scan-and-label algorithm illustrated on a simple image.

3. The line corresponding to the bridge pixel must have unit-support (pixel support of only one, i.e. the bridge pixel must not previously be linked to any other edge pixel). This constraint ensures that the bridge pixel does not participate in more than one line leading to intersecting lines. Since object boundaries do not normally intersect and our objective is to detect physical object boundaries as lines, this is a valid constraint.

4. The edge-directions that the bridge pixel can have are restricted by the edge-direction of the label pixel. These directions are specified by templates as described later in this section.

When all constraints are satisfied the algorithm does the following:

1. The bridge pixel is assigned the same label as the label pixel. The line record corresponding to its line label is reset and the line label is saved for future assignment to another edge pixel.

2. The current pixel is also assigned the line label of the label pixel. The end-point slot of the line is set to the coordinates of the current pixel. The pixel-support and average-contrast slots of the line are appropriately updated.

The effect of this procedure on Figure 3.2(a) is shown in Figure 3.2(b) (both the bridge pixel and the current pixel acquire the line-label A). For further clarity, Figure 3.3 illustrates this scan-and-label process on a simple edge image.

Continuity is tested by heuristic templates that were derived after observing several images. The set of templates for edge directions 1 and 2 are shown in Figures 3.4 and 3.5. This template matching can be done in parallel. A

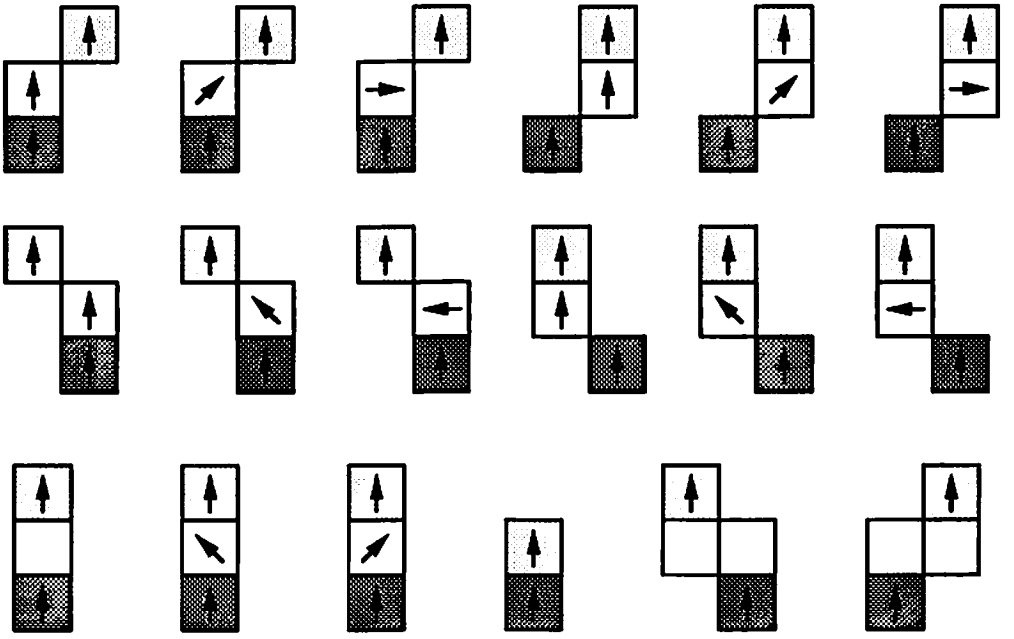


Figure 3.4: Templates for edge direction 1.

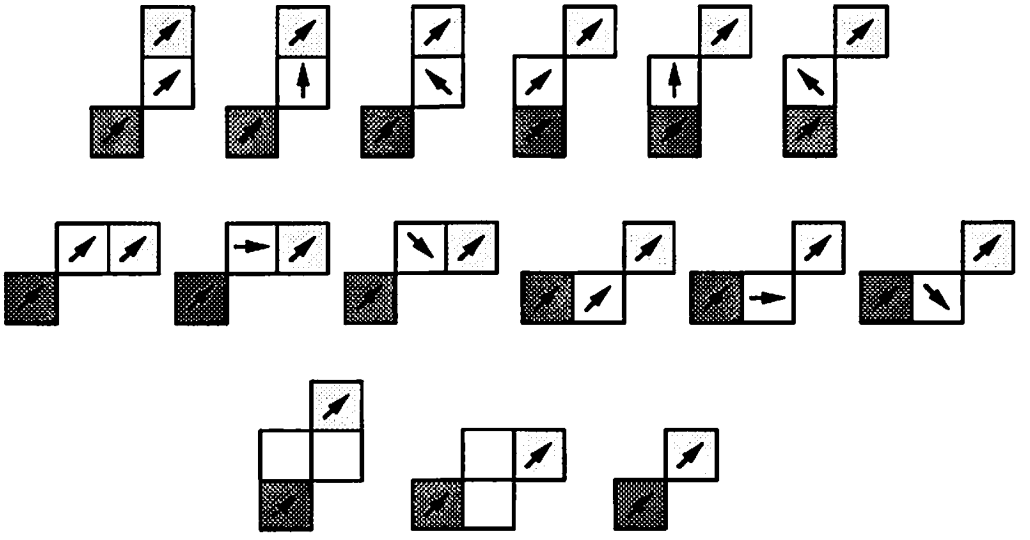


Figure 3.5: Templates for edge direction 2.

random choice can be made in the case of a conflict (virtually unlikely in the case of reasonable quality images, as our experience shows). As can be seen from Figure 3.4, these template patterns are natural: the current pixel and the label pixel have the same direction and the bridge pixel maintains spatial and directional continuity.

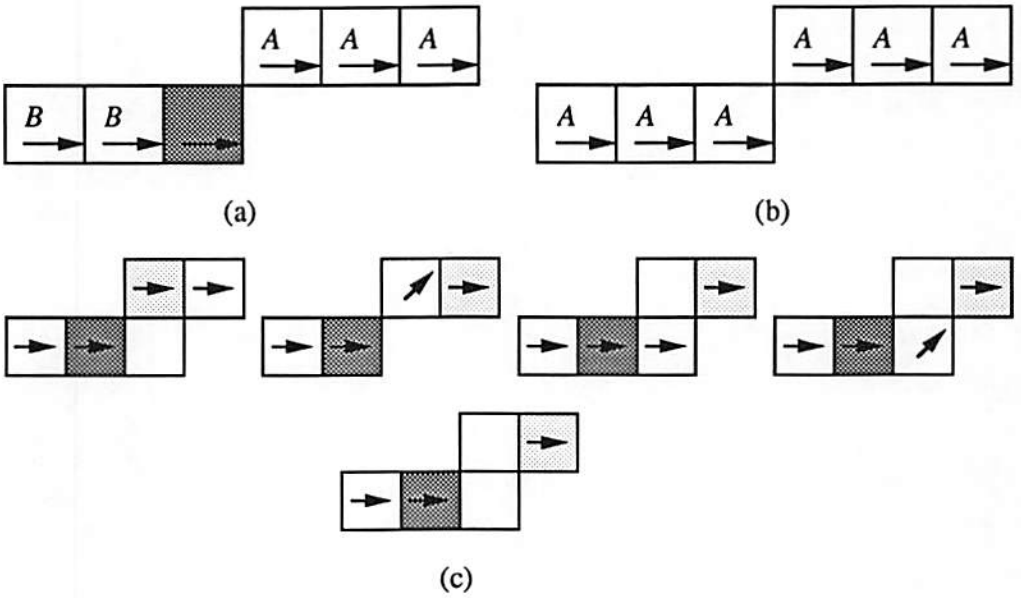


Figure 3.6: (a) One possible scenario when the current pixel direction is 3. (b) Correct labelling. (c) Special case templates for direction 3.

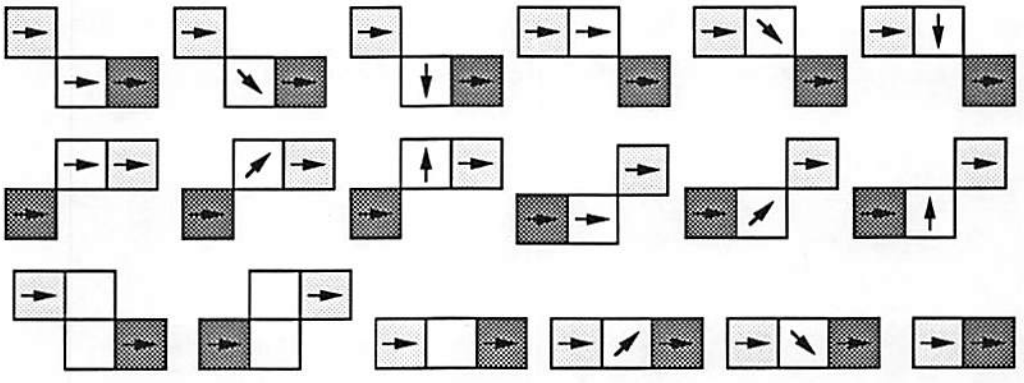


Figure 3.7: Templates for edge direction 3.

Special case: Consider the special case where the current pixel direction is along the scan direction (direction 3 in Figure 3.6. Direction 7 is also along the scan direction and its case is analogous to that of direction 3). The situation shown in Figure 3.6(a) is frequently encountered. In this case, the line-label A is given to the current pixel. Also, those pixels with line-label B are relabelled as A (Figure 3.6(b)). The templates in Figure 3.6(c) will check for such situations. Other templates for direction 3 are shown in Figure 3.7.

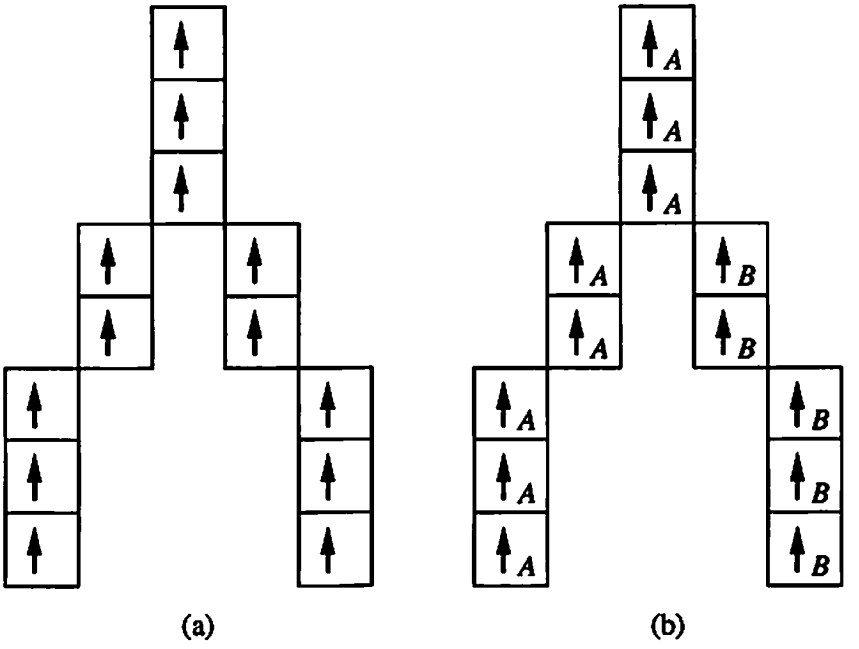


Figure 3.8: (a) A fork. (b) Line labels assigned.

Forks: Forks, like that shown in Figure 3.8(a), rarely occur in images. The labelling algorithm described in this section assigns a unique line label to each edge pixel and thereby fragments the fork shown in Figure 3.8(a) into the two lines A and B shown in Figure 3.8(b). Other types of forks will be similarly fragmented.

At the end of the LRTB scan, each edge pixel has a line label associated with it. Edge pixels that belong to the same line 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. A list of all the line labels assigned (*line-label-list*) is also accumulated. As an illustration of the effects of the scan-and-label algorithm, consider a small window of the edge image of the aerial image shown in Figure 3.18. This window is shown in Figure 3.11. Figure 3.12 shows the line labels assigned to the edge pixels after the scan and label algorithm.

3.3 Linking Unit-support Lines

The Canny edge detector is a step edge detector (as are all edge detectors in popular use) and so does not work work well at corners where two edges meet. The reason is that at the corners the image is not a perfect step and this leads to a different response than in the interiors of lines where the image is a closer approximation to a perfect step. The effective result is that the scan-and-label process often misses linking the edge pixels at the ends of lines. These edge pixels are fragmented into unit-support lines. So another stage was devised mainly to repair the effects of this fragmentation.

We observed the patterns formed by these unit-support lines over several images and as in previous stage coded these as templates. However there are fewer templates: 16 in all. Six of those (for edge directions 1, 2 and 3) are shown in Figure 3.9. The templates for the other directions are similar to these. The image is again scanned LRTB. All the unit-support lines in the current window are first marked. At each unit-support line we examine the neighborhood to see

if merging is possible with another unit-support line or with a line composed entirely at this stage of unit-support lines. Merging involves revising the slots of the record corresponding to the label pixel and marking the line label of the current pixel with the label of the label pixel and removing it from the line-label-list. Another scan then uses the marks to update the label image. Figure 3.13 illustrates the effect of implementing this stage on Figure 3.12. All the labels have changed because our system currently optimizes label assignments by reassigning labels (from 1 to the total number of lines).

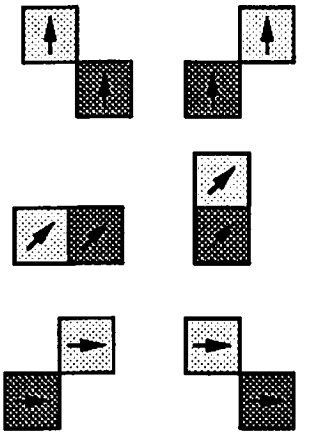


Figure 3.9: Templates for linking unit-support lines.

Even after this merging operation, some unit-support lines remain. Some of these lines have no physical significance whatsoever and are pure noise. However, a small percentage of them are natural extensions of some longer line. So we examine each line in the line-label-list that has a pixel-support greater than 1. At the end points of these lines, based on the line direction we identify the pixel coordinates within a 5×5 window that are natural extensions to the line. We then check if there are any unit-support lines at these coordinates. If the edge pixel direction is within ($\pm 1 \bmod 8$) of the quantized line direction, then the

long line is extended and merged with the unit-support line. At the end of this stage any remaining unit-support lines are treated as noise and removed from the line-label-list. The label image is appropriately updated. The effect of this stage on Figure 3.13 is shown in Figure 3.14.

3.4 Merging Collinear Lines

Real images will be noisy, corrupted by occlusions, camera distortions, surface markings, varying surface reflectance, quantization, etc. The net result is that many of the lines extracted do not correspond to object boundaries and those that correspond to object boundaries can be fragmented. Our objective is to link the fragmented line segments together and eliminate those lines that do not correspond to object boundaries. The current section is concerned with the former task. Section 3.5 deals with the task of eliminating noisy lines.

In the previous stages the decisions to link were based on small local neighborhoods (Figures 3.4, 3.5, 3.7 and 3.9). This results in an efficient implementation, but it also leads to fragmentation of straight object contours that are interrupted by noise or poor contrast. A fourth stage was devised to link these collinear fragments together.

Different techniques can be used to link collinear lines. Using the Hough transform [93] method, each line can be indexed into a quantized θ-ρ plane (θ is the orientation of the normal to the line and ρ is the perpendicular distance of the line from the origin). Large clusters in this quantized space then correspond to a fragmented line. There are several drawbacks to this approach. It does not directly take the proximity of line segments into account. For lines that are far

from the origin even a small error in θ leads to a large error in ρ. Since small errors in θ are common, collinear lines that are far from the origin are frequently left unmerged. Also, large clusters can occur at random, especially because the quantization process has the effect of integrating the noise by collecting random events within a bucket [96]. So we chose not to use Hough clustering.

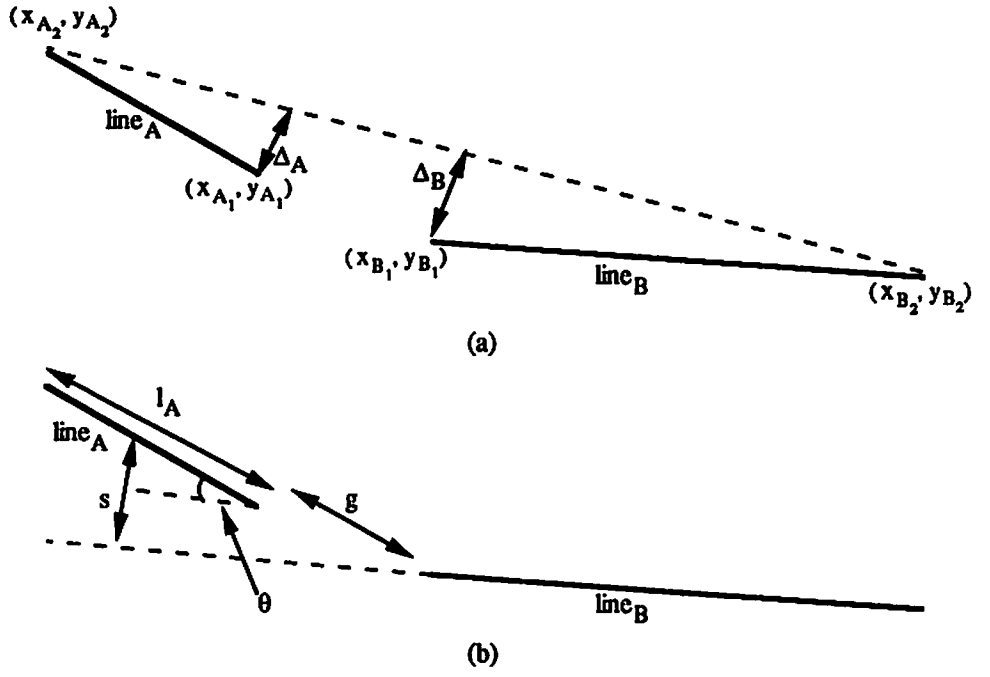


Figure 3.10: (a) Merging collinear lines. (b) Lowe's collinearity measure.

The label image can be used as a spatial index to search for pairs of proximate collinear lines. For each line in the database (say, $line_A$), we examine a 7×7 neighborhood on the label image centered at its start point. The sequence of steps to identify all of its potentially collinear lines is listed below:

1. All the line labels in the neighborhood are accumulated in a list.
2. An end point of each line in this list must lie within the neighborhood of search, otherwise the line is deleted from the list.

3. As a coarse filter only those lines in the list whose direction is within 30° of $line_A$ are retained. (A direction field is also associated with each line record and all the line directions are computed and stored before the start of this stage. This direction can be from 0° to 360° and is computed from the end-points so that the region to the right of the line is brighter than that to the left).

4. We then join the farthest end-points of the pairs of lines in each match to form a hypothetical line and ensure that the sum of the maximum absolute deviations of each line from this hypothetical line is less than a threshold. This is illustrated in Figure 3.10(a), where the total deviation $\Delta_A + \Delta_B$ must be less than a collinearity-deviation-threshold (τ_d) for the lines $line_A$ and $line_B$ to be merged. Only those lines that pass this test are retained.

5. To resolve any conflicts in the resultant matches we use Lowe's collinearity significance measure [17] to rank these matches and choose the one with the maximum measure. This measure computes the significance of collinearity between lines. For example; the smaller the angular difference, the higher the significance. In Figure 3.10(b) the significance is proportional to $\frac{{l_A}^2}{\theta s(g+l_A)}$, where l_A is the length of the smaller line, g is the gap size, θ is the difference in orientation and s is the perpendicular distance of the longer line from the mid-point of the shorter line. The previous step may seem unnecessary because Lowe's measure can be used immediately. But our experience shows that Lowe's measure can lead to very non-intuitive results when used by itself (for example: high significance even when $\Delta_A + \Delta_B$ is

a large quantity, if the line lengths are large); however, it is very effective at ranking the matches passed by the previous step.

If at the end of these steps there is a candidate match, it is merged with the current line. This linking proceeds recursively until there are no more matches. The procedure is repeated for the other end point of the line as well. The effect of linking the collinear lines in Figure 3.14 is shown in Figure 3.15.

3.5 Suppressing Noisy Lines

All the lines that are extracted by the straight line extractor will be asserted in a database for manipulation by the high level module. The smaller the number of lines, the more efficient the high level module will be. The objective of the line extractor presented here is to detect the straight object boundaries of man-made objects in images. So ideally it must extract only those lines. However, we find that a significant number of lines extracted (greater than 50%) do not correspond to object boundaries. How do we get rid of noisy lines so that the search space of the higher level module is drastically reduced? We first need to characterize the noisy lines in some way and then delete the lines that meet these characteristics. Perfect characterization is, however, not possible. So some true object edges could also be deleted (and some noisy lines remain undeleted). But the savings in execution time in the higher-level modules may well be worth a few mistakes lower down.

Based on our observation of several images, we have developed some criteria to get rid of noisy lines.

1. All lines that are shorter than a certain length ($\tau_{l_{small}}$) are deleted. This is because most of the short lines in the image do not correspond to any object boundary.

2. All lines whose average contrast is less than a threshold τ_c are deleted. The idea is that physical object edges exhibit a reasonably strong contrast. However, those lines with average contrast less than τ_c, but whose length exceeds $\tau_{l_{long}}$ are not deleted. Since long edges have strong physical significance, this will help retain some object boundaries that failed the contrast criteria.

All the lines in the line-label-list are tested with these two criteria. The label image is then appropriately updated. The effect of this noise suppression step over the label image in Figure 3.15 is shown in Figure 3.16.

After this initial stage of noise suppression many noisy lines still remain. Consider short line fragments in the image. A short line fragment has a high probability of being noise. If the fragment corresponds to a portion of an object boundary we can expect to find other fragments near its end-points. So if we delete all those short line fragments that have no other fragments close to their end-points, we can eliminate more insignificant lines from the database. In our implementation we delete all those line fragments that are less than a certain length τ_{l_1} and do not have the end-point of a longer fragment (length $> \tau_{l_2}$) within a 7×7 neighborhood of both of their end points. Fortunately, this is efficiently implemented, because this neighborhood is quickly accessible from the label image. Other line detection methods that do not have a data structure

corresponding to the label image will not be so efficient. This stage when used on the label image in Figure 3.16 results in the label image in Figure 3.17.

3.6 Real Image Examples

We present three real image examples to illustrate our algorithm. These are images of Los Angeles Airport (LAX) (320×320 pixels), the Pentagon (512×512 pixels) and Moffett Airfield (500×500). The resolutions (distance between the pixels) and the imaging conditions differ. Consequently the thresholds used during processing had to be optimized separately for each image. If the images had been taken under the same conditions the thresholds would be the same.

LAX image. Figure 3.18(a) shows an image of a portion of LAX airport. Figure 3.18(b) shows the result of applying the Canny operator. The dark dots correspond to edge pixels. For the Gaussian convolution the standard deviation (σ) used was 0.7. For hysteresis thresholding, a low threshold of 50 and a high threshold of 200 were used [102]. The scan-and-label procedure when applied to the edge image in Figure 3.18(b) results in the detection of the 6220 lines shown in Figure 3.18(c). After linking unit-support lines, extending long lines through collinear unit-support lines and deleting any remaining unit-support lines, the number of lines is reduced to 5791 (Figure 3.19(a)). The 2201 lines that result after merging collinear lines are shown in Figure 3.19(b) ($\tau_d = 1.5$). After a thresholding operation based on the line length and contrast, the 1128 lines shown in Figure 3.19(c) result ($\tau_{l_{small}} = 1$, $\tau_c = 37$ and $\tau_{l_{long}} = 8$). After the deletion of isolated short line fragments, the 847 lines shown in Figure 3.20(a) are obtained ($\tau_{l_1} = 5$

and $\tau_{l_2} = 7$). Together the two noise suppression steps reduce the number of lines from 2201 to 847. This order of reduction is typical and leads to significant savings in memory as well as computational time for the higher level modules. For comparison purposes we show the outputs obtained by the Nevatia-Babu method (Figure 3.20(b)) and the Burns method (Figure 3.20(c)). The thresholds were tuned manually to get the best possible results.

Pentagon image. This image (Figure 3.21(a)) is a good candidate for testing the algorithms as it has straight line contours in many directions. Figures 3.21 and 3.22 show the sequence of steps when detecting the lines ($\tau_d = 2.0$, $\tau_{l_{small}} = 2$, $\tau_c = 49$, $\tau_{l_{long}} = 8$, $\tau_{l_1} = 7$ and $\tau_{l_2} = 11$). The difference in thresholds from the previous example is because the images were taken under different conditions and resolution. Figure 3.23(a) shows the final result. Figure 3.23(b) shows the results obtained by the Nevatia-Babu method and Figure 3.23(c) shows those obtained by the Burns method.

Moffett Field image. Figures 3.24 and 3.25 show the sequence of steps when detecting lines on an aerial image of Moffett Airfield. The final result is shown in Figure 3.26(a). ($\tau_d = 2.3$, $\tau_{l_{small}} = 4$, $\tau_c = 56$, $\tau_{l_{long}} = 7$, $\tau_{l_1} = 9$ and $\tau_{l_2} = 11$). Figures 3.26(b) and (c) show the results obtained by the Nevatia-Babu and Burns methods respectively.

A qualitative comparison based on visual inspection shows that the line detector presented in this chapter gives good results. The issue of computation time is, however, more difficult to resolve. These algorithms run on different processors, are written in different languages and are in different stages of development

(including code optimization, etc). We have observed that the elapsed time is about the same for the three methods compared. As a case study consider the Pentagon image. Our algorithm implemented in LISP on a Texas Instruments Explorer LISP machine with 16MB memory took approximately 18 minutes to detect the lines. The Nevatia-Babu method implemented in C on a SUN-3/260 with 24MB memory took approximately 18 minutes. The Burns detector implemented in C on a SUN-3/260 with 16MB memory took approximately 25 minutes for the entire processing. The Burns line detector results in the detection of finer features but is considerably more noisy. This is partly the result of the design philosophy used: the smallest possible masks that will produce estimates of the gradient direction were chosen, as the primary goal of the line detector is the recovery of lines corresponding to fine detail. The small masks often lead to a noisy response.

3.7 Discussion and Extensions

We have presented a straight line extractor for detecting physically significant straight line contours of man-made objects from their 2D projections. Only a small (at most 7×7) neighborhood is accessed at any time and this leads to an efficient implementation. This algorithm was primarily designed for serial computers. On a parallel computer some of the stages should be changed to take advantage of the parallelism (for example, a parallel connected components algorithm must be devised for the second stage). However, the basic design philosophy will remain unchanged.

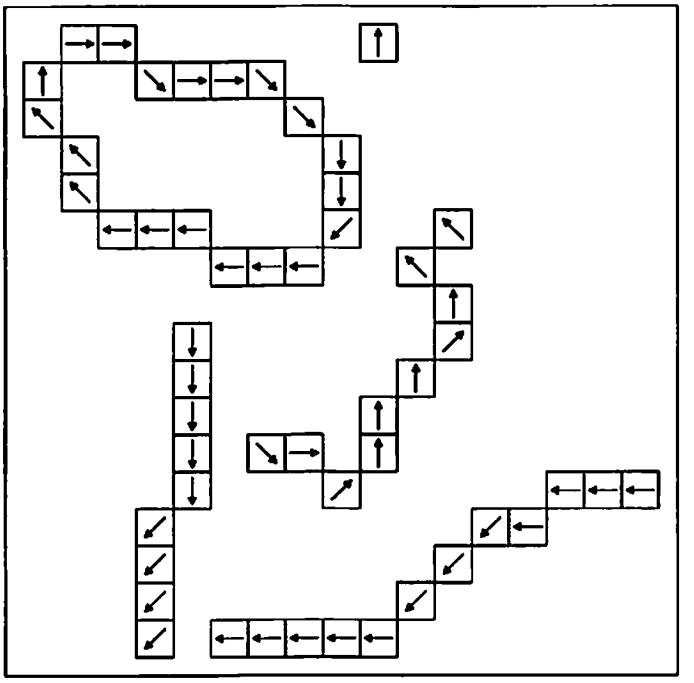


Figure 3.11: A sub-window of an edge image.

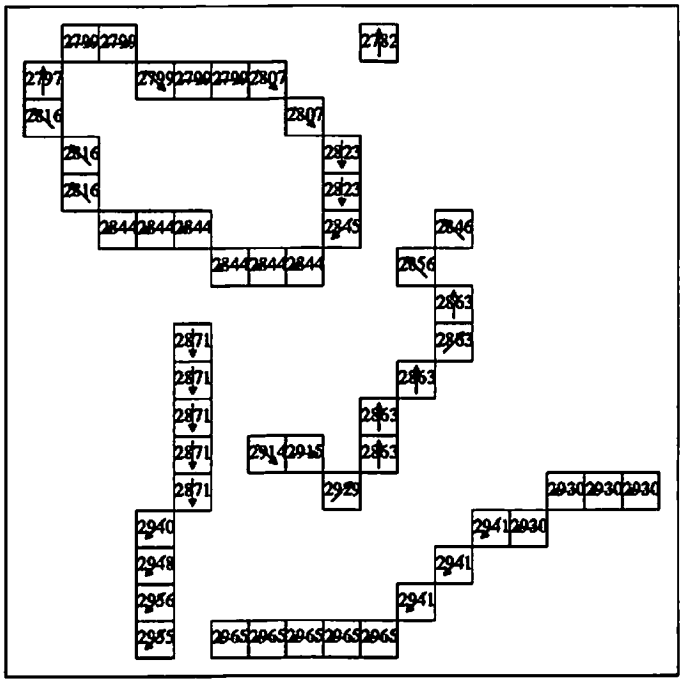


Figure 3.12: Labels assigned after the scan-and-label process.

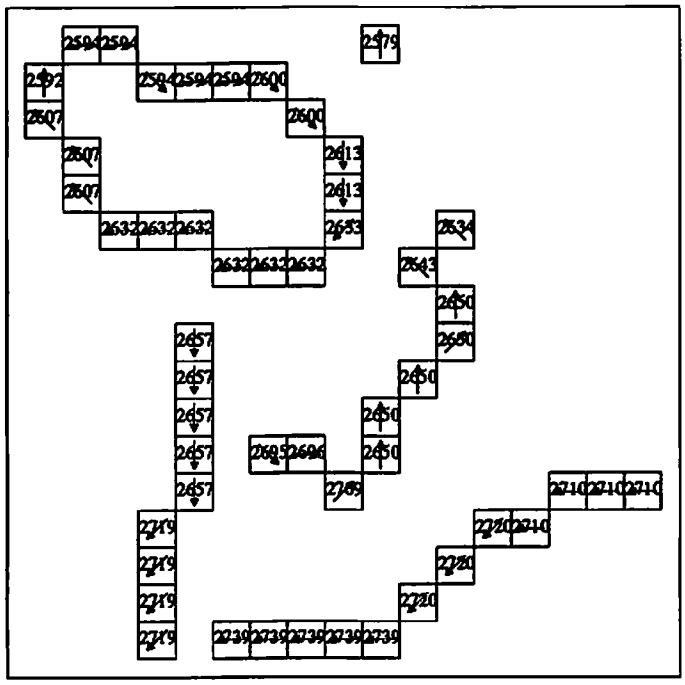


Figure 3.13: Labels assigned after linking unit-support lines.

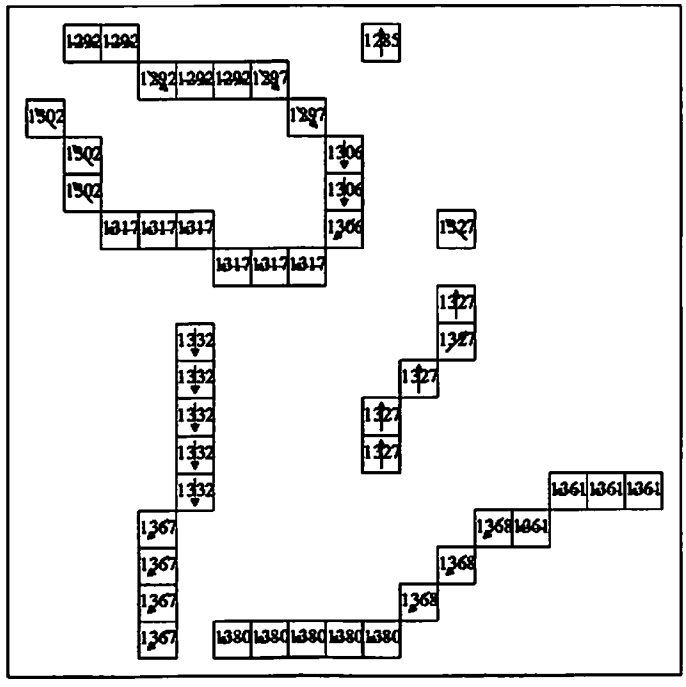


Figure 3.14: Labels assigned after extending the lines.

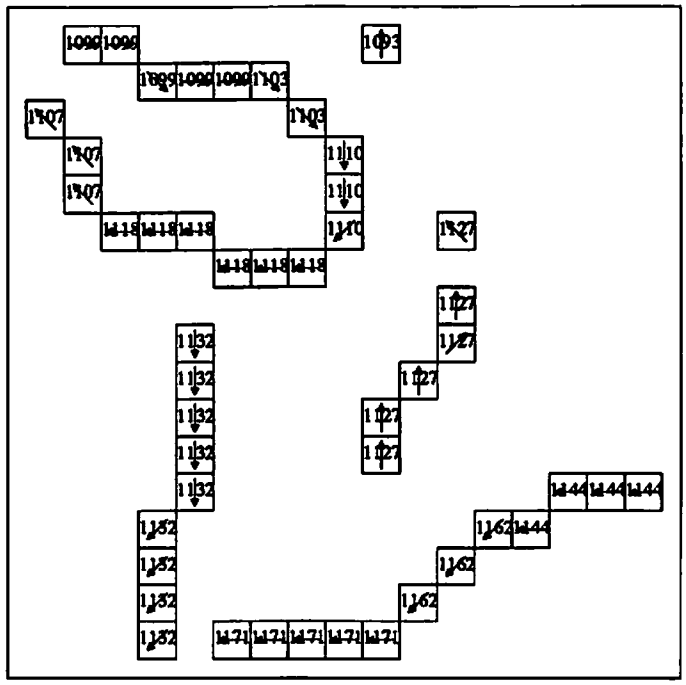


Figure 3.15: Labels assigned after linking collinear lines.

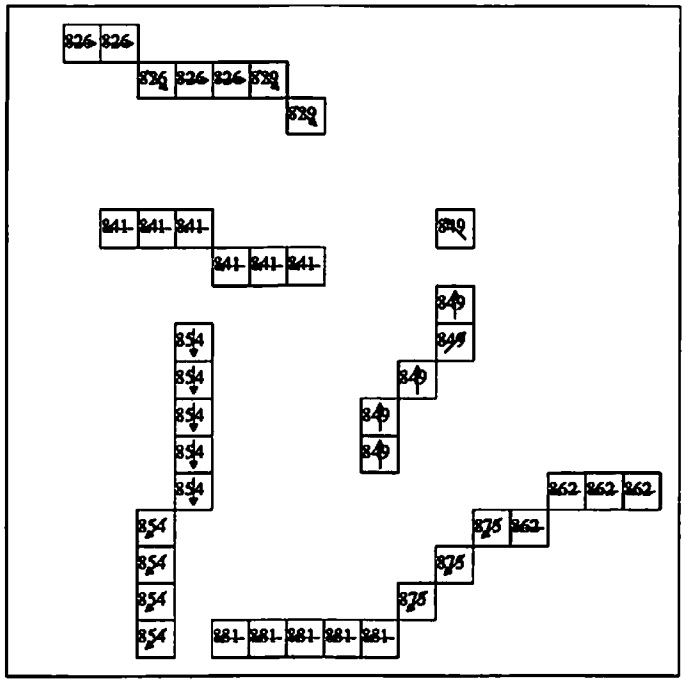


Figure 3.16: Labels assigned after thresholding for noise suppression.

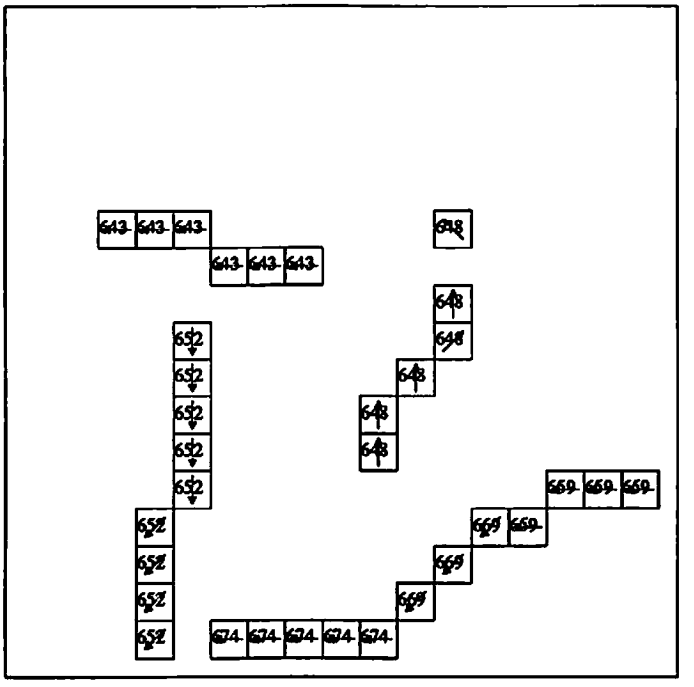
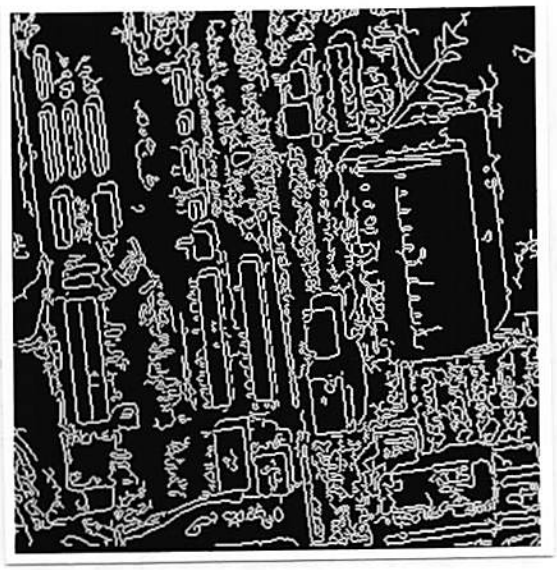


Figure 3.17: Labels assigned after deleting isolated short lines.

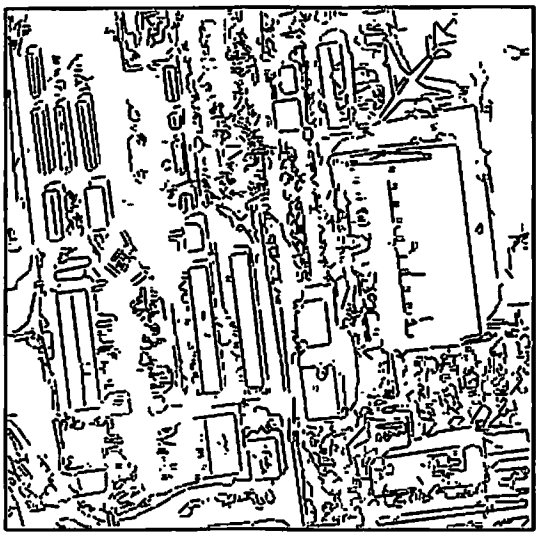
(a)



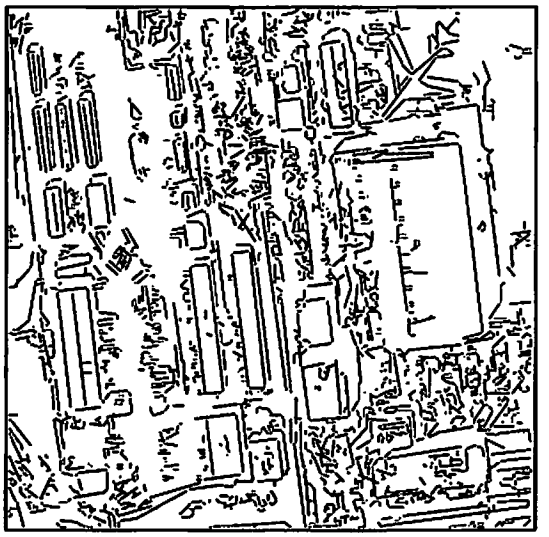
(b)

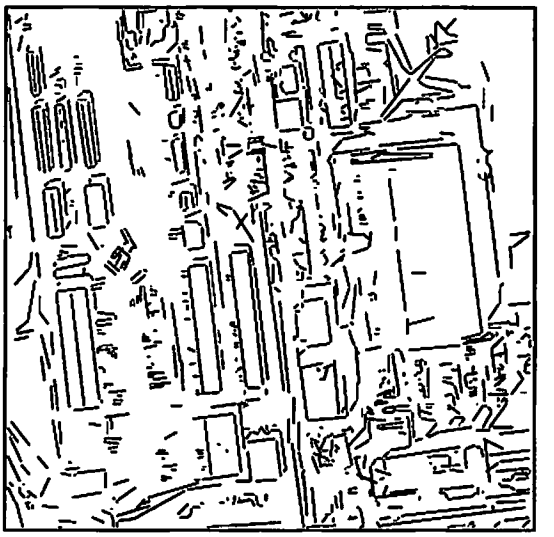
(c)

Figure 3.18: (a) LAX image. (b) Edges detected. (c) Set of lines after the scan-and-label algorithm.



(a)

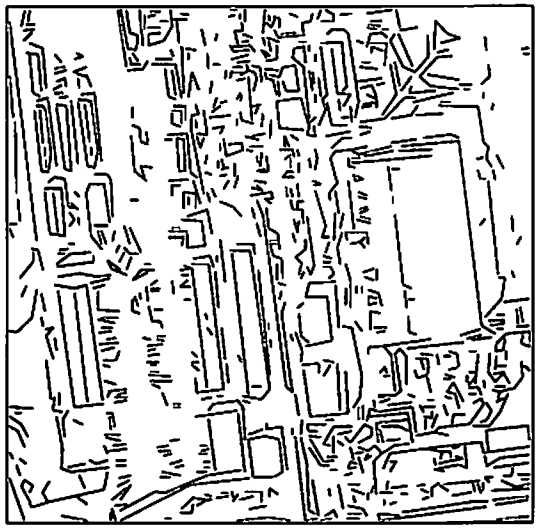


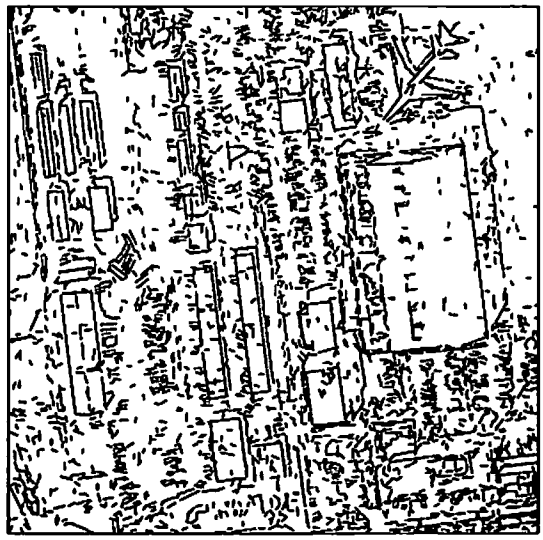


(b) (c)

Figure 3.19: Set of lines after (a) linking unit-support lines, (b) merging collinear lines, (c) thresholding on length and contrast.

(a)



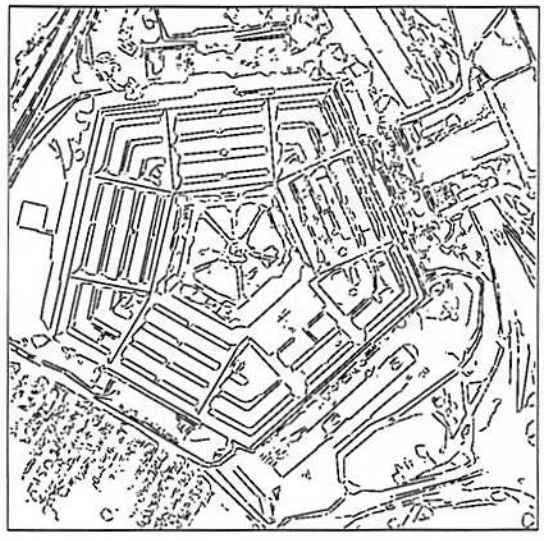


(b) (c)

Figure 3.20: (a) Final set of lines obtained by our method. (b) Lines obtained by Nevatia-Babu method. (c) Lines obtained by Burns method.

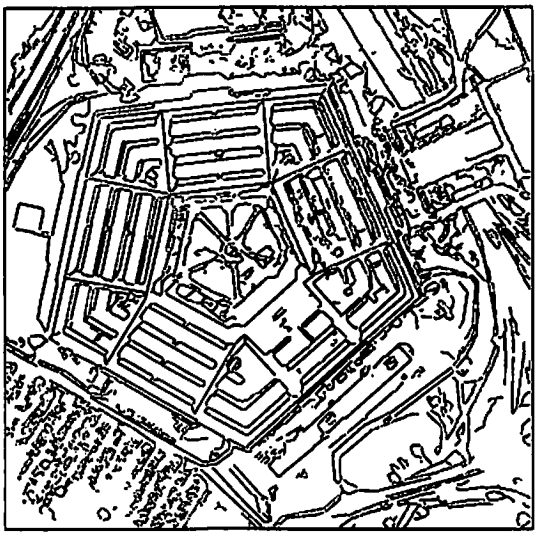
(a)

(b)

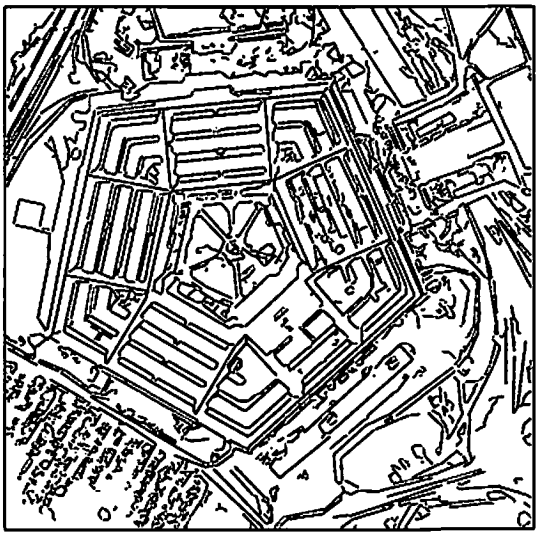


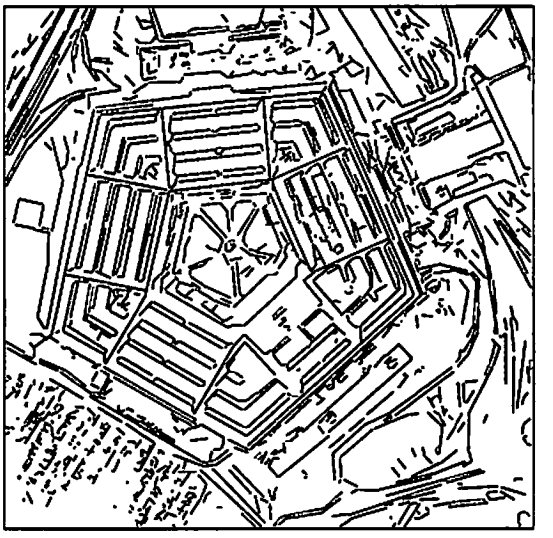
(c)

Figure 3.21: (a) Pentagon image. (b) Edges detected. (c) 9016 lines after the scan-and-label algorithm.



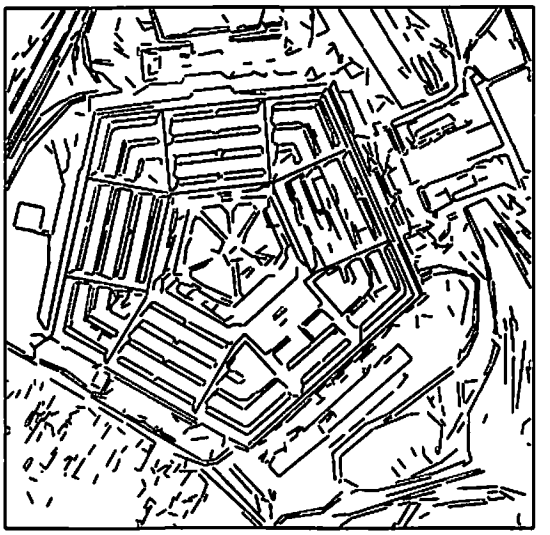
(a)



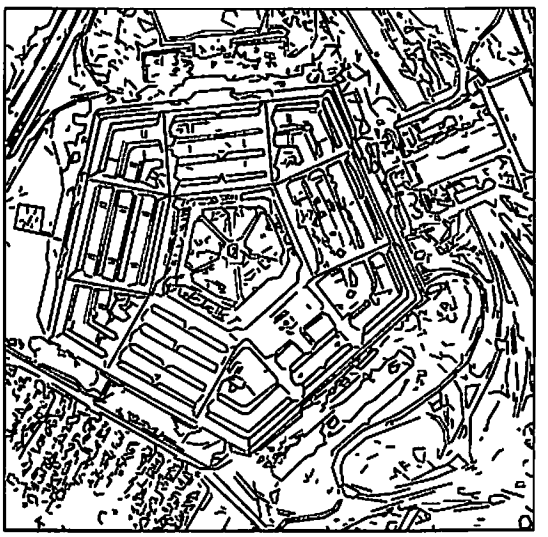


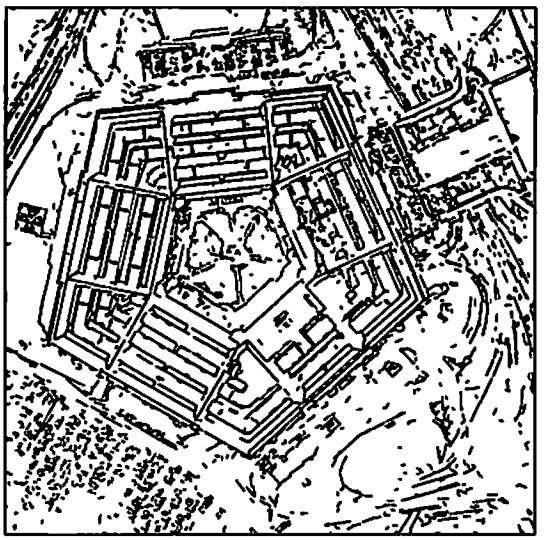
(b) (c)

Figure 3.22: (a) 4797 lines after linking unit-support lines. (b) 3082 lines after merging collinear lines. (c) 1625 lines after thresholding on length and contrast.



(a)



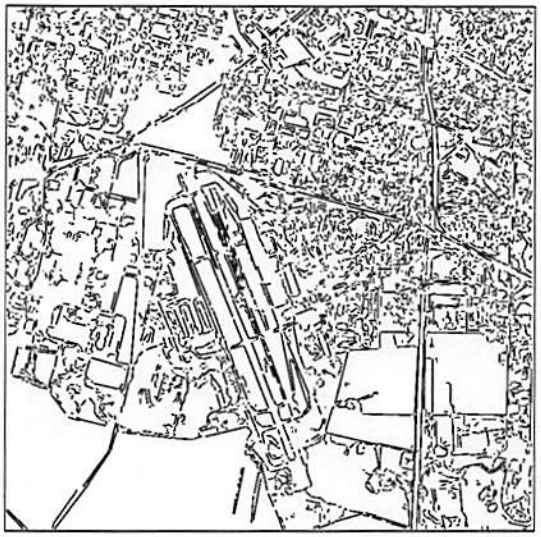


(b) (c)

Figure 3.23: (a) Final set of 1209 lines obtained by our method. (b) Lines obtained by Nevatia-Babu method. (c) Lines obtained by Burns method.

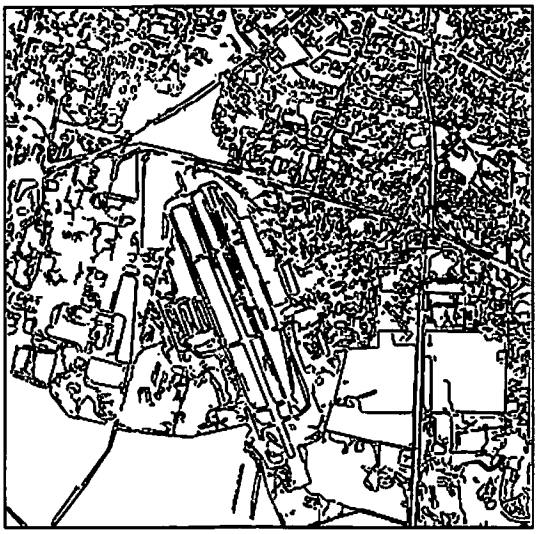
(a)

(b)

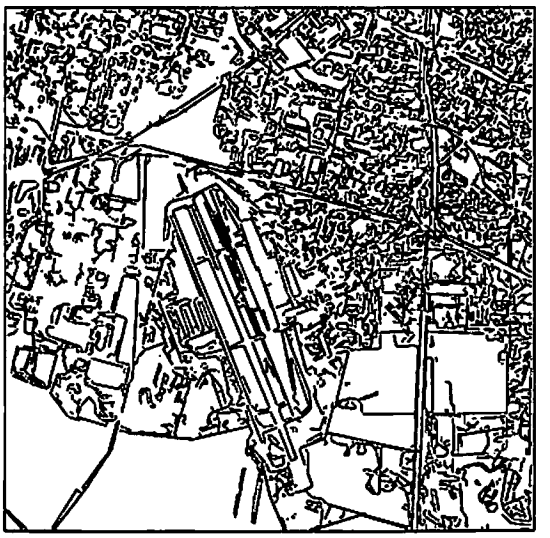


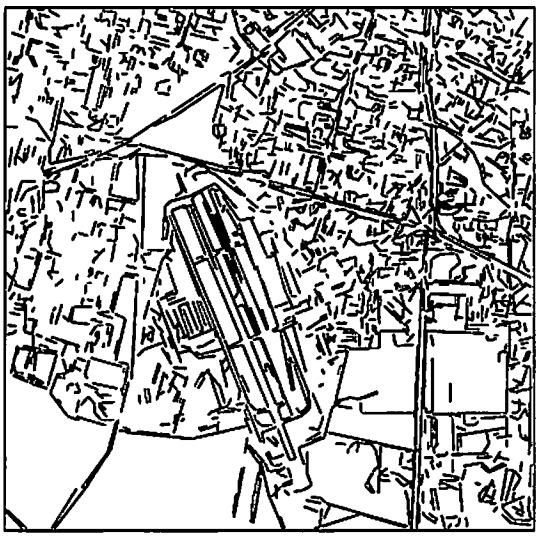
(c)

Figure 3.24: (a) Aerial image of Moffett field. (b) Edges detected. (c) 17396 lines after the scan-and-label algorithm.



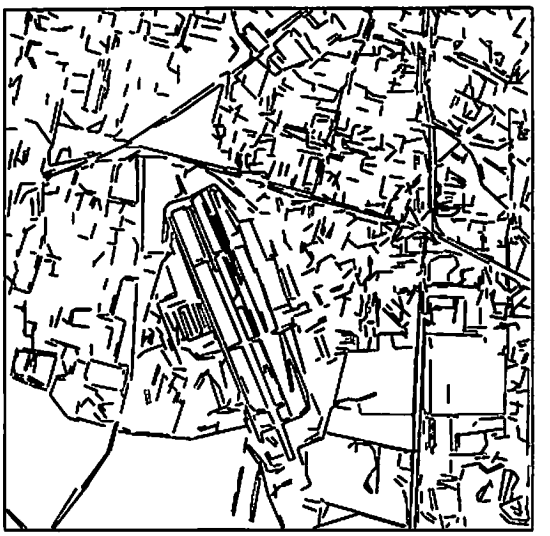
(a)



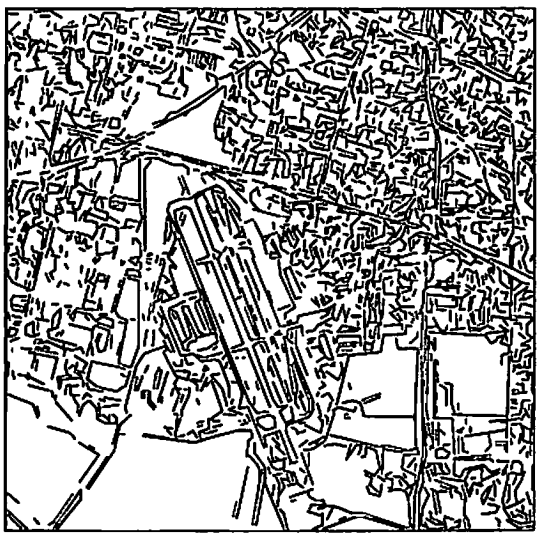


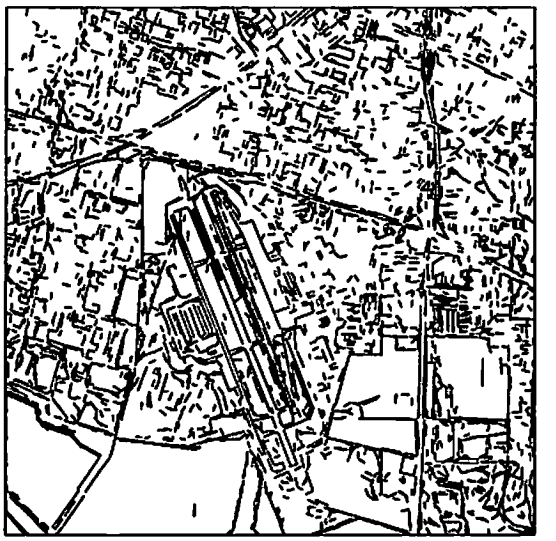
(b) (c)

Figure 3.25: (a) 8151 lines after linking unit-support lines (b) 5707 lines after merging collinear lines (c) 1979 lines after thresholding on contrast and length



(a)





(b) (c)

Figure 3.26: (a) Final set of 1408 lines obtained by our method. (b) Lines obtained by Nevatia-Babu method. (c) Lines obtained by Burns method.

Chapter 4

Detection of Buildings in Aerial Images

4.1 Introduction

We present a system that can detect buildings in monocular aerial images and also reconstruct their shape. The class of buildings modeled have roofs with orthogonal corners and the walls are assumed to be vertical. The process of object synthesis is carried out in a hierarchical manner. First potential *vertices* in the image are identified by looking for orthogonal and proximate line segments. Vertices are then grouped together to hypothesize roof *edges*. Edges are composed into *edge-rings*. Roof hypotheses are generated for *closed-rings* with shadow evidence. Some of the edge-rings will only be partially complete. Heuristics based on the knowledge of the structure of buildings are used to hypothesize missing elements top-down to close some these edge-rings. An ATMS is used to integrate bottom-up and top-down searches. For each confirmed roof hypothesis, a surface

based description of a building is generated. The walls of the buildings are assumed to be perpendicular to the ground plane. This surface based description can be rendered from any viewpoint, either as a wireframe or as a gray level image. One contribution of this work to the area of building detection is that we explicitly address the issue of alternatives during the search process. We also present a dynamic search strategy that can deal with incremental addition and deletions of hypotheses through the use of truth maintenance. We also address the issue of efficient computation of geometric relations during the grouping process given a noisy environment.

4.2 Hierarchical Feature Grouping

The buildings in our system are modeled to have polygonal roofs with orthogonal corners and the walls are assumed to be vertical. Similar models have also been used in other building detection work [11, 18, 64]. The input to the system is in the form of line segments. The Canny operator [102] is first used to get an initial set of edge pixels. These edge pixels are then linked [104] to form a set of line segments. The line segments produced by the low level module are asserted into the database as *instances* of a generic line-segment frame. This constitutes the starting state of the database. Then a heuristic set of rules operates over this database and hierarchically composes the line segments into buildings.

Consider Figure 4.1(a). An ideal roof image could look like this. The dark region corresponds to the shadow cast by the building. An ideal line detector would extract the line segments shown in Figure 4.1(b). However, these examples ignore two problems. First, a real image is never as perfect as shown in Figure 4.1(a).

The image will be noisy; corrupted by occlusions, camera distortions, surface markings, varying surface reflectance, quantization etc. Furthermore, shadows do not necessarily fall on level ground. Second, the line detector will not be perfect and will introduce its own distortions. The net result is that the line segments extracted are fragmented, displaced and over-extended. Low contrast edges may go completely undetected. Figure 4.1(c) shows the kind of line segments we could expect to detect. The objective is to detect the buildings given such a noisy initial state. We use a divide and conquer strategy. We start with the line segments and hierarchically compose them into buildings, with meaningful geometrical entities represented at different levels of the hierarchy. The hierarchical composition process is heuristic in nature and is specified by using production rules.

Initially, orthogonal and proximate line segments in the image are identified to detect potential *vertices.* Vertices are then grouped together to generate *edges.* Edges are then grouped to form *edge-rings.* Closed edge-rings can correspond to roofs. Shadow analysis is used on *closed-rings* to identify roofs. Given that the initial data is noisy, there is a grouping ambiguity when going from one level of the hierarchy to the next. At any level, some elements will be missing and some will be erroneous. This uncertainty results in a search space that needs to be explored. Each object in the hierarchy is then an *assumption* that could be true or false. An ATMS is ideally suited to sift through such a search space spanned by assumptions. Missing elements in this hierarchy can be hypothesized incrementally in a top down manner, since the ATMS is completely incremental. We have chosen to use the ATMS only at the sparsest level of the hierarchy: the edge level. The reason is that, fewer the number of assumptions, the more

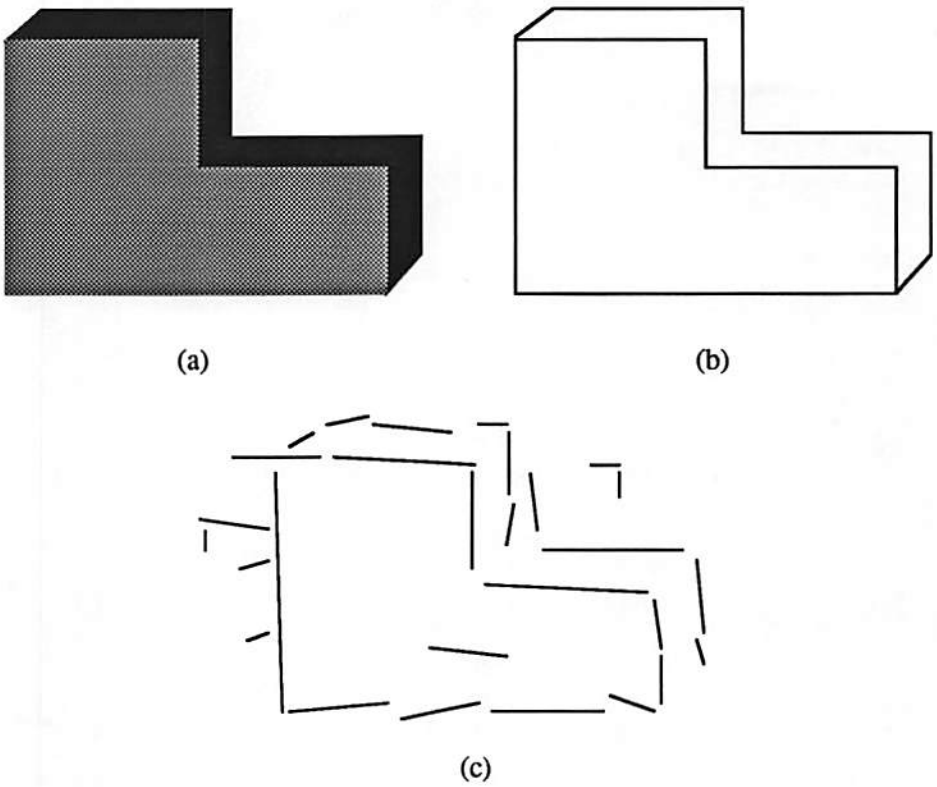


Figure 4.1: (a) An ideal roof image. (b) Line segments extracted from (a) by an ideal line detector. (c) Typical line segments from a real image.

efficient the ATMS is. Moreover, no direct useful intra-level constraints exist at the line-segment and vertex level. Thus by restricting the *assumptions* to the edge level we have tried to gain considerable efficiency at the sacrifice of an insignificant amount of completeness.

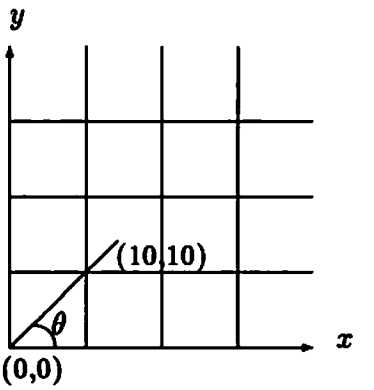
Because the input is noisy, there is an uncertainty in geometrical parameters like angles and positions of the objects in the hierarchy. So at each stage of the composition process we need some thresholds to limit the number of objects in the next level of the hierarchy. The challenge is to provide a uniform framework for such decisions. We modeled the uncertainty in the angles as user specified numerical value Δ_θ and the uncertainty in position as Δ_λ. The uncertainty in the

gray levels of the image was modeled as Δ_g. We tried to reduce all thresholding to a comparison test between certain geometrical (or image) parameters and these numerical values. We define two operators $\ominus$ and $\vec{\ominus}$ that measure angular differences between two vectors. $\vec{\ominus}$ measures the inside angle between the two vectors and so always returns a value less than 180°. $\ominus$ considers each vector as a line and measures the acute angle of bisection. So it returns a value less than 90°.

Frames are used to represent all the objects and their relations in this hierarchy. An object oriented programming paradigm is used. Operations over objects are performed by passing *messages* to the objects. *Active values* defined over some frame slots are triggered asynchronously by certain operations over the slots, and usually take care of side effects of some main inference action. Each frame has four kinds of slots: *normal*, *parameter*, *coarse filter* and *relation*. *Normal* slots store the values of a set of geometrical parameters that completely describe the frame. *Parameter* slots are used to store the values of certain parameters that would otherwise be repeatedly computed, leading to wasted time. *Coarse filter* slots serve as coarse filters for limiting comparisons when computing relations between objects. Quantized angles or distances are stored in these slots. Symbolic relations between frames are stored in *relation* slots.

As an example of a frame, consider the line segment frame, *line-12*, in Figure 4.2. Consider the parameter slots ω_x, ω_y and ω_0, which are precomputed and stored. This helps in quick geometric computations. For example, the perpendicular distance of a point (x_0, y_0) from this line is quickly computed as: $| \omega_x \cdot x_0 + \omega_y \cdot y_0 + \omega_0 |$. Consider the coarse filter slots Θ-bucket, Ω_0-bucket and *Grid*-bucket. The entire angular range of 0° to 180° is quantized into Θ-buckets.

line-12 frame		
slot-type	*slot*	*value*
Normal	start	(0 0)
	end	(10 10)
	contrast	34
Parameter	length	14.141
	θ	45.0°
	ω_x	0.707
	ω_y	-0.707
	ω_0	0.000
Coarse filter	Θ-bucket	5
	Ω_0-bucket	0
	Grid-bucket	(0 0)
	Grid-bucket	(0 1)
	Grid-bucket	(1 0)
	Grid-bucket	(1 1)
Relation	instance-of	line-frame
	line-of	limb-31



$\omega_x = \cos\theta$
$\omega_y = -\sin\theta$
$\omega_x x + \omega_y y + \omega_0 = 0$
Θ-bucket width = 10°
Ω_0-bucket width = 10
Grid-bucket dimensions
= (6.67 6.67)

Figure 4.2: A line frame example.

Similarly the image is divided into a grid as *Grid*-buckets. The perpendicular distance from the origin, ω_0 is quantized into Ω_0-buckets. This quantization process is illustrated in Figure 4.3. The purpose of this quantization is to reduce the number of comparisons when computing relations. For example, to detect all pairs of parallel line segments we can first restrict the search to line segments that index into same or adjacent Θ-buckets. To identify collinear line segments, we can further restrict the search to line segments that index into adjacent or same Ω_0-buckets, since collinear lines have the same perpendicular distance from the origin. Similarly features are indexed by their location into a spatial grid, as in Figure 4.3(b). This will be useful in proximity detection.

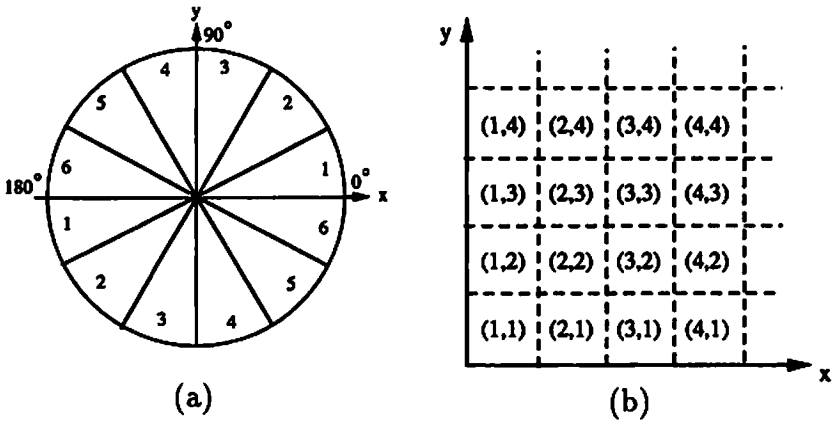


Figure 4.3: Bucketing techniques are used to reduce comparisons. (a) Angle buckets. (b) Grid buckets.

4.2.1 Vertices

Consider now, the process of detecting vertices. A rule detects potential vertices by searching the database for orthogonal and proximate line segments and then checking that some constraints are satisfied. The geometry is illustrated in Figure 4.4.

- coarse filters:

  The coarse filter slots of the line segment frame are used to reduce this search space. We first collect those pairs of line segments $(line_A, line_B)$ that index into orthogonal Θ-buckets and also index into same or adjacent *Grid*-buckets.

- $|90 - \theta| < \Delta_\theta$

  This constraint checks for perpendicularity.

- $d_1 + d_2 < \Delta_\lambda$

  This constraint checks for proximity.

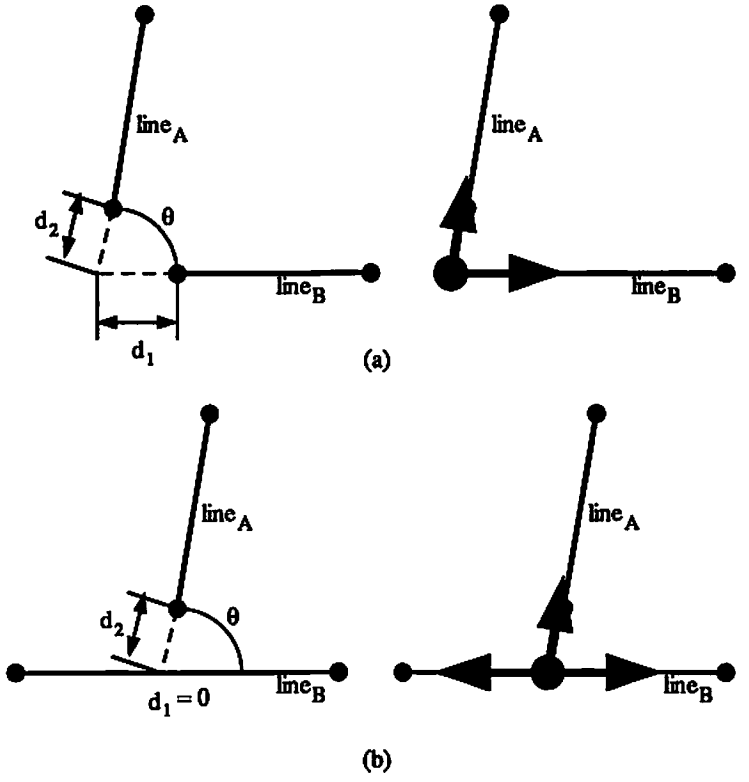


Figure 4.4: Vertex geometry.

The result of executing this rule on Figure 4.1(c) is shown in Figure 4.5. 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 line segments forming the vertex.

A vertex-frame has a normal slot for its coordinates. It further has a parameter slot for its *orientation*. To calculate the effective orientation of a vertex, for each limb of the vertex, we determine the angle in the first quadrant that is 45° or 135° away. The average of all such angles of the different limbs of the vertex is called the orientation of that vertex. For example, all the vertices shown in Figure 4.6 have the same orientation $\vec{O}$. Coarse filter slots for a vertex frame index the *Grid*-bucket its coordinates fall in and the Θ-bucket its orientation

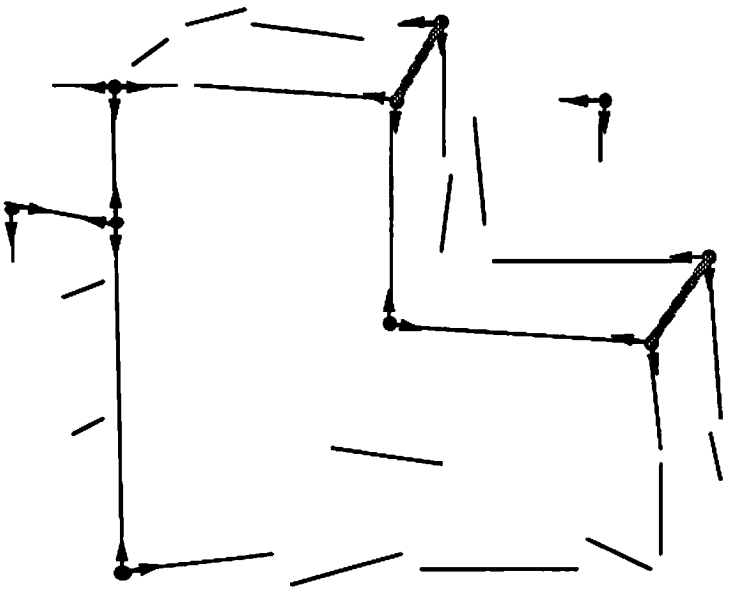


Figure 4.5: Vertices

falls in. A limb-frame is very similar to a line segment frame. Its vector orientation is specified by an angle between 0 to 360°. It is linked to the vertex that it belongs to by the *limb-of* relation, and the line segment that it is aligned with by the *has-line* relation. *Has-limb* is the inverse of *limb-of* and *line-of* that of *has-line*. A limb *inherits* its coordinates from the vertex it is connected to, through the *has-limb* relation. and the the parameters ω_x, ω_y and ω_0 along the

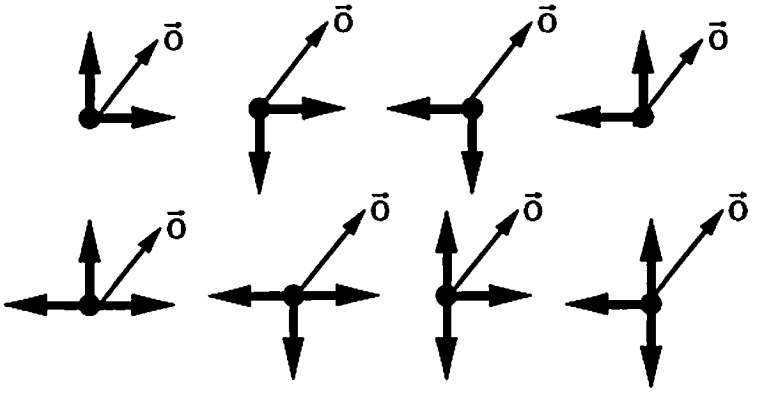


Figure 4.6: Vertex orientation

line-of relation. The advantage of inheritance mechanisms like this is that they are wired in and so they are specified outside of the production rules. This has the effect that the production rules are easier to read as side-effects like this are hidden from the specifications. This has the effect of making program revision much easier.

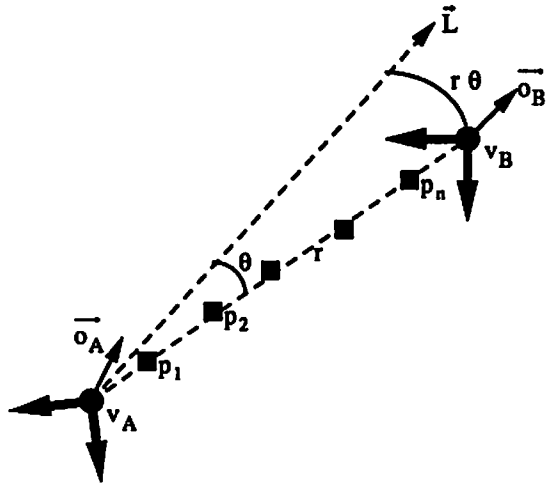


Figure 4.7: Shadow vertex geometry

It is assumed that the illumination vector is known. This is used to identify vertices that are in shadow correspondence. For two vertices to be in shadow correspondence the line joining the vertices must have the same direction as the projection of the illumination vector on the ground plane. The geometry and the constraints are illustrated in Figure 4.7, where $\vec{L}$ is the projection of the illumination vector along the ground plane.

- coarse filters:

  We choose those pairs of vertices (v_A, v_B) whose orientations index into same or adjacent Θ-buckets.

- The illumination vector must be at an acute angle with two limbs of the object vertex and the shadow vertex must have limbs that are in the same direction as these limbs, within the angular uncertainty Δ_θ.

- $\vec{O_a} \ominus \vec{O_b} < \Delta_\theta$.

- $\theta < \Delta_\theta$. This measures the angular deviation from true correspondence.

- $r\theta < \Delta_\lambda$. $r\theta$ measures the positional deviation from true correspondence.

- The image is sampled at regular intervals at $p_1, p_2, \ldots, p_n$ along the line joining v_A and v_B. To ensure that this line is in a shadow region, the mean gray level of these points must be less than a user specified gray level. The standard deviation must be less than Δ_g, the uncertainty in gray level.

Figure 4.5 shows the two shadow correspondences that were derived using these constraints. Gray lines in this figure connect the vertices that are in shadow correspondence.

4.2.2 Edges

If two limbs are coincident and are facing each other, then their corresponding vertices can potentially form an edge. This is a robust technique for detecting object edges, because object edges are frequently fragmented into small line segments. The geometry for detecting edges is illustrated in Figure 4.8.

- coarse filters:

  We choose those pairs of limbs (l_A, l_B) that index into same or adjacent Θ-buckets and Ω_0-buckets.

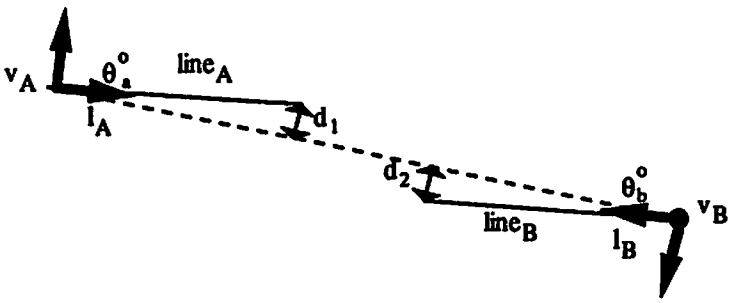


Figure 4.8: Edge geometry

- $\left|180 - (\theta_A \vec{\ominus} \theta_B)\right| < \Delta_\theta$

  This constraint ensures that the limb vectors are approximately 180° apart.

- $d_1 + d_2 < \Delta_\lambda$

  Together with the above constraints, this constraint ensures that the two limbs are approximately collinear and face each other.

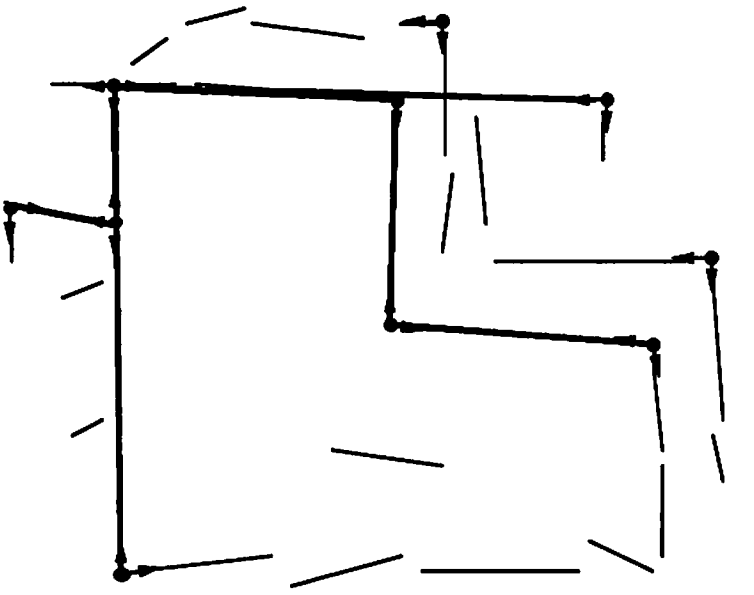


Figure 4.9: Initial set of edges

All the edges so identified from Figure 4.5 are shown in Figure 4.9. An edge frame is also very similar to a line-segment frame. It has *normal* slots for the

end points and *parameter slots* for its direction θ and also ω_x, ω_y and ω_0; the parameters of the line along the edge. *Coarse filter* slots index it into Θ, Ω_0 and *Grid*-buckets. *Relation* slots connect it to the limbs that form it. Links are automatically placed to the vertices that form it.

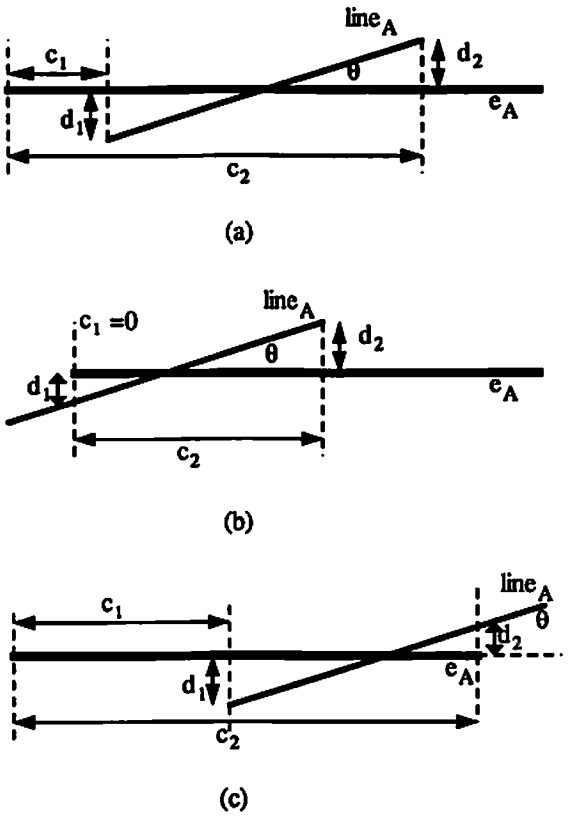


Figure 4.10: Accumulating the support of the edges

However, not all the edge frames so generated have sufficient low level line segment support. To compute the support, potentially overlapping pairs of edges and line segments are first identified. The geometry for this is illustrated in Figure 4.10, which shows three scenarios of overlap.

- coarse filters:

  Look for edge and line segment pairs $(e_A, line_A)$ that index into same or adjacent Θ, Ω_0 and *Grid* buckets.

- $\theta < \Delta_\theta$

- $d_1 + d_2 < \Delta_\lambda$

- finite support: c_1 and c_2 are distances measured from one end vertex of the edge along the edge and as such can be negative.

  If c_1 or c_2 is negative set it to 0.

  If c_1 or c_2 is greater than the edge length l_A of edge e_A, then set it to l_A.

  Now we test to see that the projection is finite, i.e.

  $c_2 - c_1 > 0$.

If these constraints are satisfied, the projection interval (c_1, c_2) is stored in the *support* slot of the edge frame. This is a multiple valued slot and any number of such intervals can be stored. A rule then examines these support slots and deletes the edge frame if the support is insufficient to warrant an object edge hypothesis. To determine if the support is adequate, overlapping intervals are merged and the total effective support is computed as a percentage of the edge length. The largest gap in the support is also computed. An edge is deleted if the percentage support is less than a certain threshold or the largest gap in the support is more than a certain threshold. These thresholds serve to reduce the number of edges participating in the edge ring composition process. Some true edges could be deleted and some false edges remain undeleted. But the edge ring composition process to be described later is sufficiently robust to deal with these

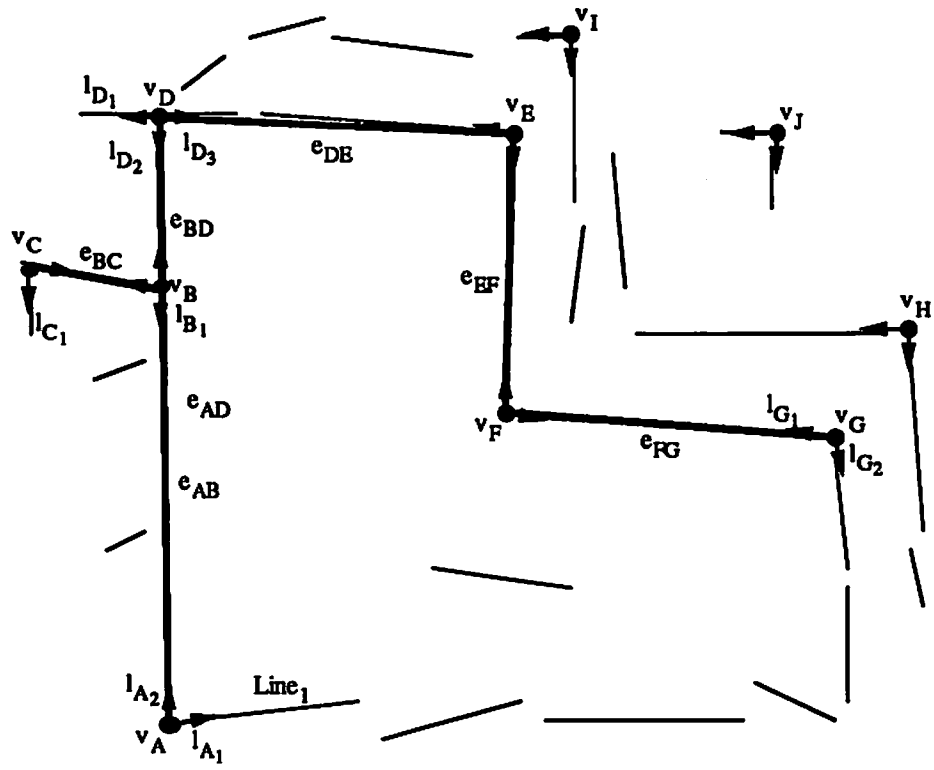


Figure 4.11: Set of edges after deleting edges with insufficient low level evidence.

mistakes. As such these thresholds are not very crucial. In Figure 4.9, one edge fails to meet these thresholds and is removed from the database. The result is shown in Figure 4.11.

4.2.3 Edge relations

The next step is to compose these edges into edge-rings. To speed up this process certain relationships between the edges are precomputed and stored for fast retrieval. The *connects* relationship links two edges if they are orthogonal to one another and are connected at a vertex. The *overlaps* relation links two overlapping edges. The *intersects* relation links intersecting edges. The *touches* relation links two edges that are very close to each other, but neither intersect, overlap

nor are connected. All four relations are inverses of themselves. All these relations are used in the edge-ring composition process. The reason for computing all these relations is this: An edge-ring cannot have two edges that *intersect*, *overlap* or *touch*, because this leads to the formation of closed-rings that cannot physically correspond to roofs. The *touches* relation is to ensure that the roofs are never too narrow. Every edge in an edge-ring must be *connected* to the adjacent edge in the ring.

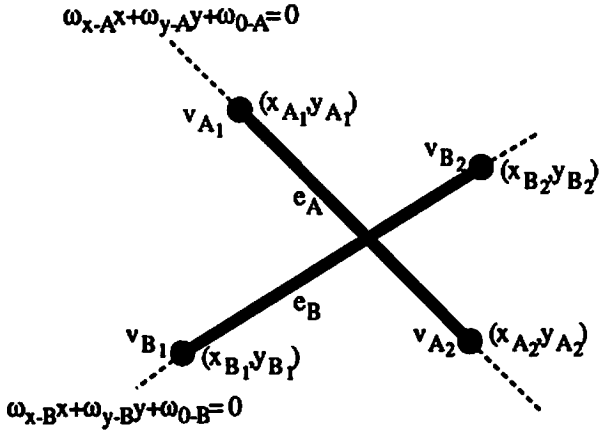


Figure 4.12: Intersecting edges

For the computation of the *intersects* relation, consider the geometry shown in Figure 4.12.

- coarse filters:

  Select those pairs (e_A, e_B) of edges that index into the same *Grid*-bucket.

- To ensure proper intersection we first need to ensure that each end point of an edge is greater than the positional uncertainty Δ_λ from the other edge.

  $|\omega_{x\text{-}B}x_{A_1} + \omega_{y\text{-}B}y_{A_1} + \omega_{0\text{-}B}| > \Delta_\lambda$

  $|\omega_{x\text{-}B}x_{A_2} + \omega_{y\text{-}B}y_{A_2} + \omega_{0\text{-}B}| > \Delta_\lambda$

$$|\omega_{x\text{-}A}x_{B_1} + \omega_{y\text{-}A}y_{B_1} + \omega_{0\text{-}A}| > \Delta_\lambda$$

$$|\omega_{x\text{-}A}x_{B_2} + \omega_{y\text{-}A}y_{B_2} + \omega_{0\text{-}A}| > \Delta_\lambda$$

- Finally, the end points of each edge must be on either side of the other edge.

  $$(\omega_{x\text{-}B}x_{A_1} + \omega_{y\text{-}B}y_{A_1} + \omega_{0\text{-}B}) \times (\omega_{x\text{-}B}x_{A_2} + \omega_{y\text{-}B}y_{A_2} + \omega_{0\text{-}B}) < 0$$

  $$(\omega_{x\text{-}A}x_{B_1} + \omega_{y\text{-}A}y_{B_1} + \omega_{0\text{-}A}) \times (\omega_{x\text{-}A}x_{B_2} + \omega_{y\text{-}A}y_{B_2} + \omega_{0\text{-}A}) < 0$$

All these constraints can be efficiently computed, since the parameter slots; ω_x, ω_y and ω_0 are precomputed and stored for each edge.

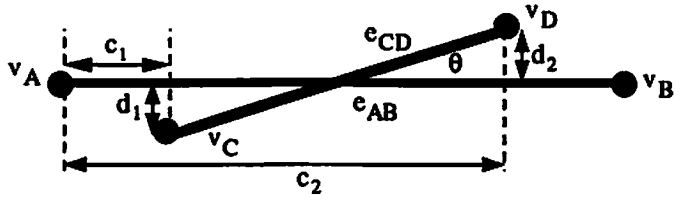


Figure 4.13: Overlapping edges

For the detection of overlapping pairs of edges consider the geometry in Figure 4.13.

- coarse filters:

  Choose those pairs of edges (e_{AB}, e_{CD}) that index into same or adjacent Θ, Ω_0 and *Grid* buckets.

- $\theta < \Delta_\theta$

- $d_1 + d_2 < \Delta_\lambda$

  d_1 and d_2 can be efficiently computed from the *parameter* slots ω_x, ω_y and ω_0. For example, in Figure 4.13,

$d_1 = |\omega_{x\text{-}AB} x_C + \omega_{y\text{-}AB} y_C + \omega_{0\text{-}AB}|$,

where (x_C, y_C) are the coordinates of vertex v_C.

- finite cover: c_1 and c_2 are distances measured from vertex v_A and can be negative. In that case they are reset to 0. If c_1 or c_2 is greater than the edge length l_{AB} of edge e_{AB}, then set it to l_{AB}.

  The overlap is finite, if now, $c_2 - c_1 > 0$.

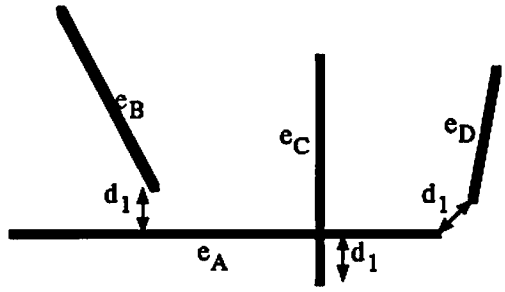


Figure 4.14: Touching edges

To detect *connected* edges we first identify the pairs of edges that share an end vertex. Then the angular difference between these edges as measured by the operator $\ominus$ must be less than Δ_θ. The *touches* relation links two edges that neither intersect, overlap nor are connected, but one of whose end point is within the position uncertainty Δ_λ of the other edge. This is illustrated in Figure 4.14, where e_A *touches* e_B, e_C and e_D if d_1 is less than Δ_λ.

4.2.4 Edge-rings

The edges and their relationships form a graph, where the edges are the nodes of the graph and the *connects*, *overlaps*, *touches* and *intersects* relations are the links. Some of these edges do not correspond to roof edges in the image and

so can be considered as clutter. Some roof edges in the image do not have corresponding edges in this graph, because they could not be detected (poor contrast, occlusion, noise in the image, imperfect detection process etc). We need to search for roofs in this noisy search space. We first assume that each *edge* is indeed a roof-edge. Some of these assumptions will ultimately be found to be false, but at this point in the search space there is no way of knowing that. The space spanned by these *assumptions* is most efficiently and naturally searched in multiple contexts for roofs by using an ATMS. The search strategy we used can be seen as a *best-contexts-first* strategy. This strategy is heuristic in nature and is specified using production rules. Specification with production rules assists in revisability and allowed us to easily experiment with different strategies till we found an acceptable one. Description of the strategy follows.

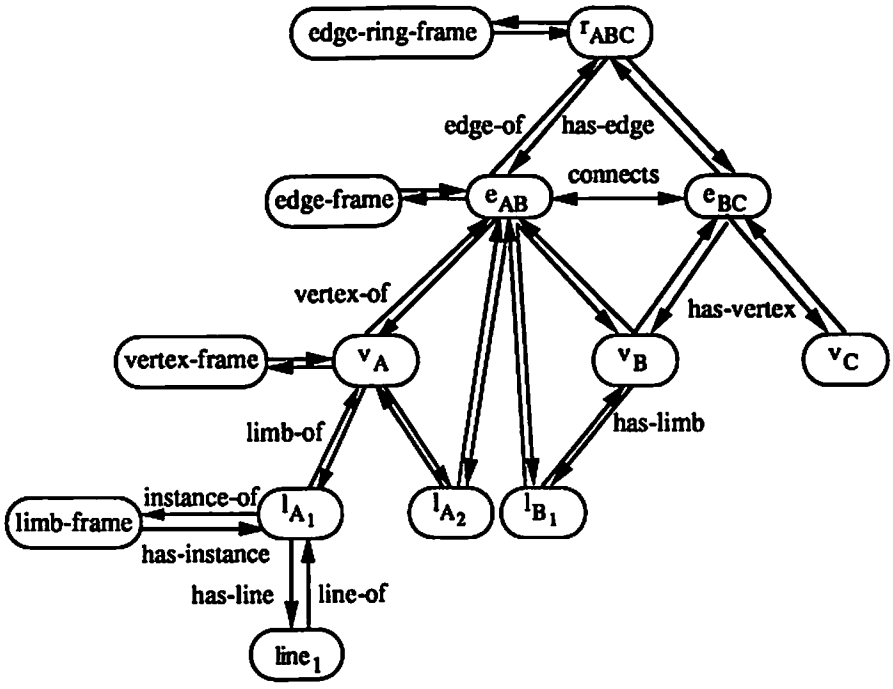


Figure 4.15: Part of the frame network

The *connects* relation is used to compose edge-rings, generating context merges in the process. The edge ring composition process is illustrated in Figures 4.15 and 4.16. Edge-rings can be composed in one of two ways. Two edges can be merged to form an edge-ring if they are linked by the *connects* relation. This is shown in Figure 4.15 where the edges e_{AB} and e_{BC} are merged to form the edge-ring r_{ABC}. This figure also illustrates part of the semantic network that is constructed by the system, with objects as the nodes and their relations as the links. Another way to form a new edge-ring is to merge an existing edge-ring with an edge that is *connected* to one of its end edges. But before they are merged it is ensured that the edge does not *overlap* or *touch* any edge in the edge-path of the edge-ring. *Intersecting* edges are stored as *nogood* environments. An *environment* of a context is defined as the set of assumptions that underly it. If this set leads to a contradictory result, it is a *nogood.* Nogoods prune the search space as no context that is a superset of the nogood is ever created by the TMS. An edge-ring with a closed edge-path is immediately identified as a *closed-ring* and it cannot participate in the composition process anymore. Figure 4.16 shows one possible scenario during the edge-ring composition process. Each node in this graph is a *context.* The *environment* of each context is shown above the line drawn through the middle of the context. The deductions based on this environment are shown below the line. The root context always has no assumptions.

An edge-ring frame has slots for *free-limbs.* The free-limb slots store the names of limbs that are attached to an end vertex of the edge-ring, but which are not part of any edge. An edge-ring that has atleast one free-limb at both end vertices is immediately identified as a *free-ring.* For example, the edge-ring r_{ABC} formed by e_{AB} and e_{BC} in Figure 4.11 has the free-limbs l_{A_1} and l_{C_1} at either

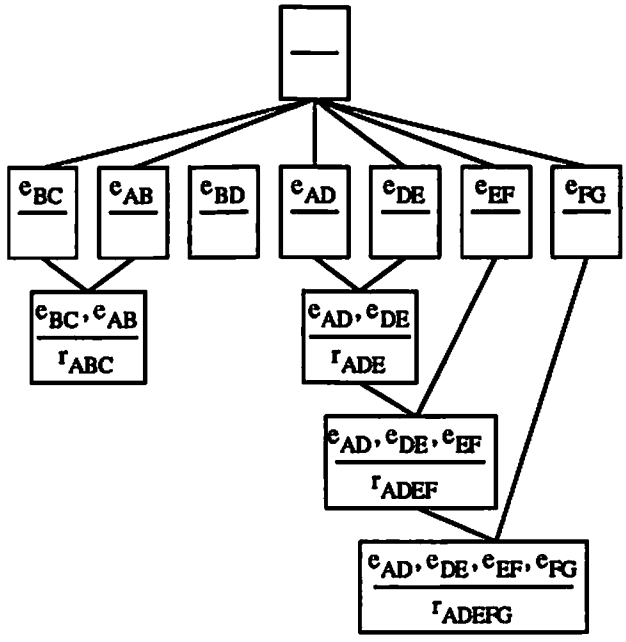


Figure 4.16: Edge ring composition.

end and so qualifies as a free-ring. In Figure 4.16, r_{ADEFG} is another free-ring. The system focuses on free-rings. Heuristics based on the knowledge of the shape of roofs are used to predict possible closures of free-rings. Currently three such heuristics are used. Consider the free-rings shown in Figure 4.17 (a), (c) and (e).

1. In Figure 4.17(a) the positions of the free-limbs l_A and l_B strongly suggests the presence of the the two edges e_C and e_D as shown in Figure 4.17(b).

2. The presence of the edge e_{AB} in Figure 4.17(d) is strongly suggested by the positions of free-limbs l_A and l_B in Figure 4.17(c).

3. The long and approximately equal length lines (length difference within Δ_λ) $line_A$ and $line_B$ of the free-limbs shown in Figure 4.17(e) can be used to hypothesize the completion of the free-ring e_{AB} as shown in Figure 4.17(f).

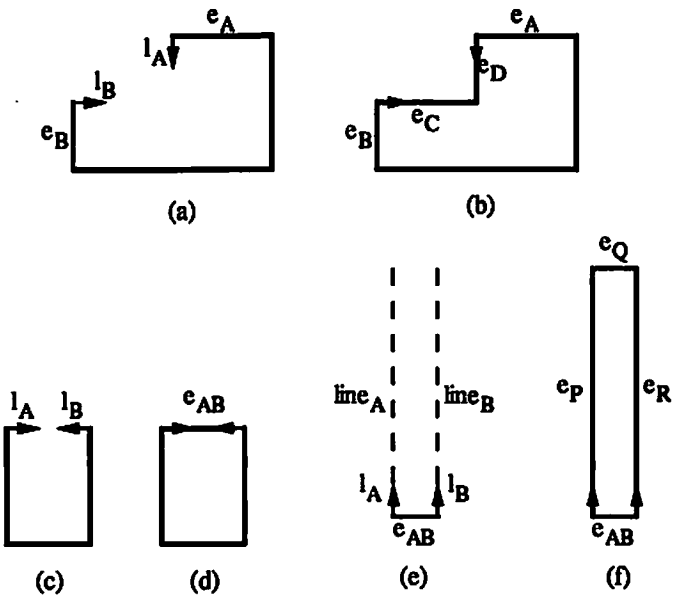


Figure 4.17: Heuristics for completing free-rings. (a) can be completed into (b), (c) to (d) and (e) to (f).

When examining the support of these top-down edges more liberal thresholds need to be used in deleting those with insufficient support. In fact, we found no need to delete any of these top-down edges at all. This shows that these heuristics are very effective. These edges now participate in the edge-ring creation process. The first heuristic applies to the graph in Figure 4.16. The new edge assumptions e_{GZ} and e_{AZ} are added top-down. The resulting situation is shown in Figure 4.18. The edge ring composition process proceeds incrementally and the situation shown in Figure 4.19 could result. $r_{ADEFGZA}$ is immediately identified as a closed-ring.

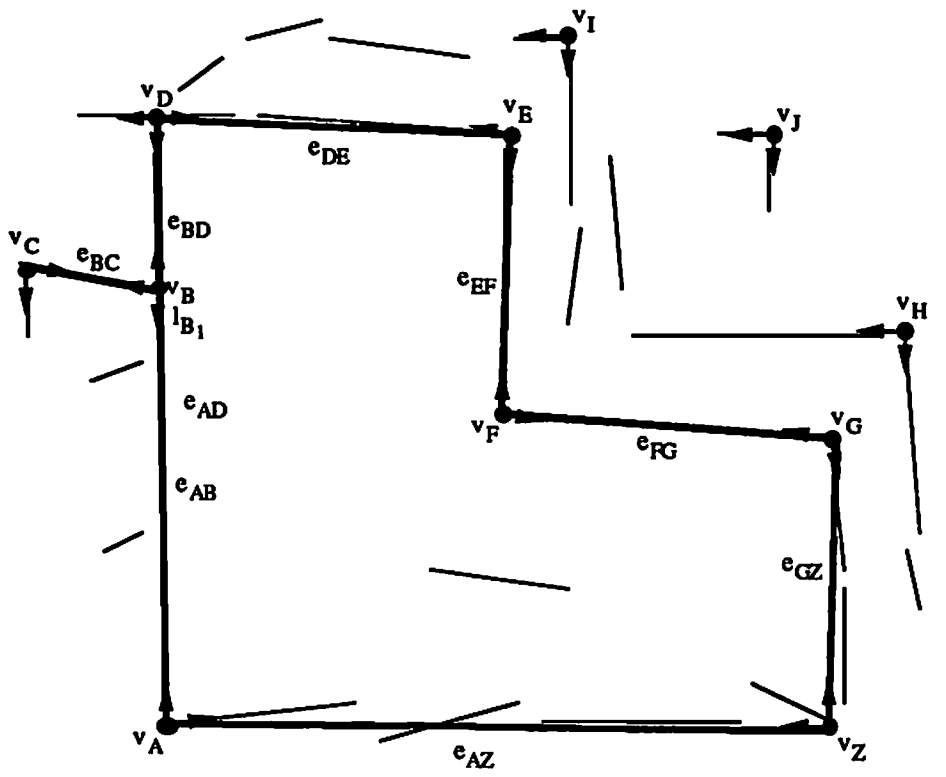


Figure 4.18: Edges hypothesized top down and bottom-up

4.2.5 Roofs

Closed-rings can correspond to roofs, so the system focuses on closed-rings and does a shadow analysis on them. If a closed ring has vertices in shadow correspondence, the correspondence with the minimum height interpretation is chosen and the closed ring is hypothesized as a roof. The height of the roof is determined by the length of the line joining the vertices in shadow correspondence and the sun angle, which is assumed to be known. In many cases, however, no vertex correspondence is available. In this case, the system needs to do a more complicated shadow analysis. The system first identifies all the shadow casting edges of the closed-ring and searches for edge-line shadow correspondence. A rule

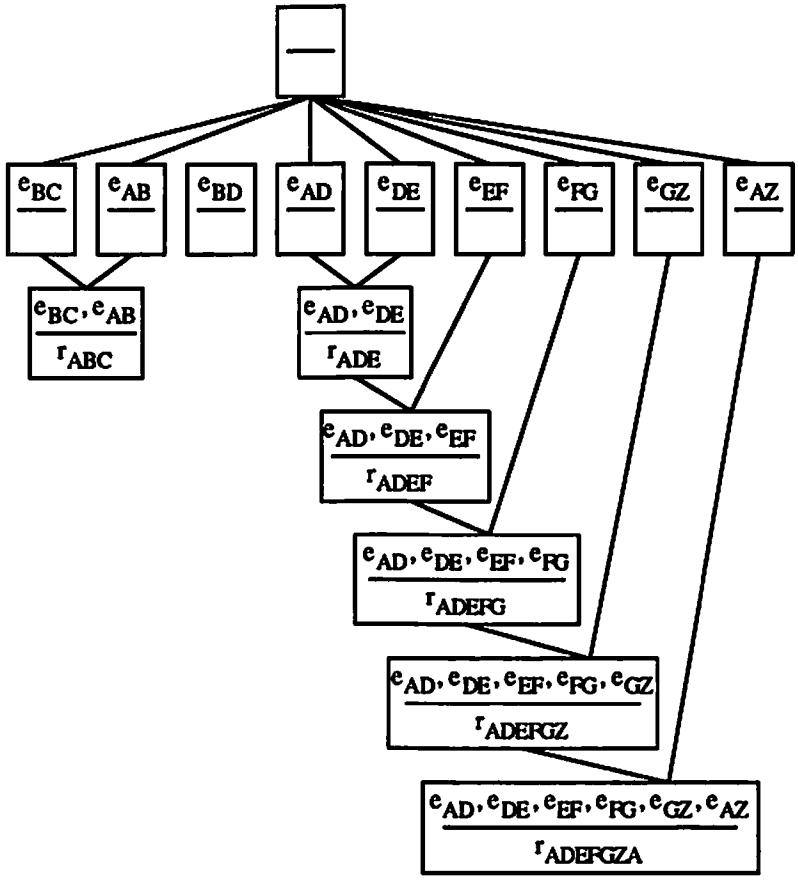


Figure 4.19: Edge ring composition after the addition of top-down edges

first accumulates all the line segments that are parallel to the edge and satisfy the constraints listed below. The geometry is shown in Figure 4.20.

- coarse filters:

  The edge, line segment pairs $(e_{AB}, line_{QR})$ that index into adjacent or same Θ-buckets are first chosen.

- length of $line_{QR}$ must be greater than Δ_λ

- The ratio of the line segment length to the edge length must be less than 1.5 but greater than 0.15. This constraint is to ensure that the edge and the line segment are of comparable length. Since the extent of fragmentation

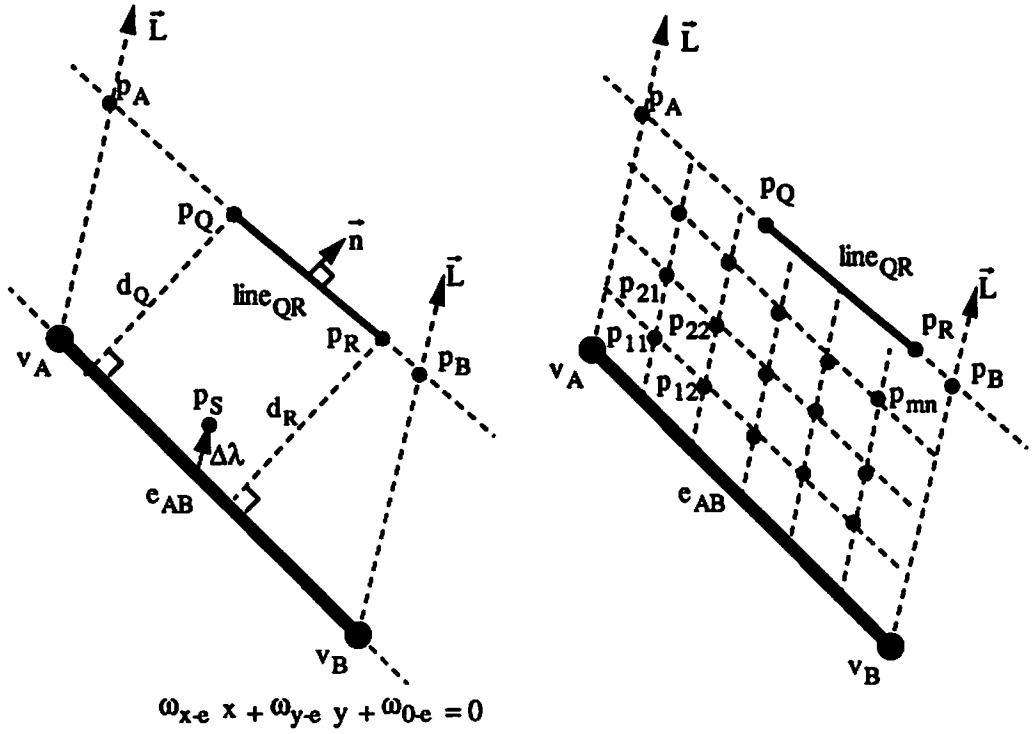


Figure 4.20: Edge, line-segment shadow correspondence.

of lines is more severe than over-extension, the threshold is more liberal in the case where the line segment length is smaller than the edge length.

- The angle difference between the line segment and edge as measured by the operator $\ominus$ is less than Δ_θ and $|d_A - d_B| < \Delta_\lambda$.

  These constraints ensure that the edge and the line segment are parallel.

- $\vec{n} \cdot \vec{L} > 0$

  $\vec{n}$ is the normal to line segment $line_{QR}$ that points in the direction of increasing brightness across the edge and $\vec{L}$ is the projection of the illumination vector along the ground. This constraint ensures that the dark region lies between the edge and line segments.

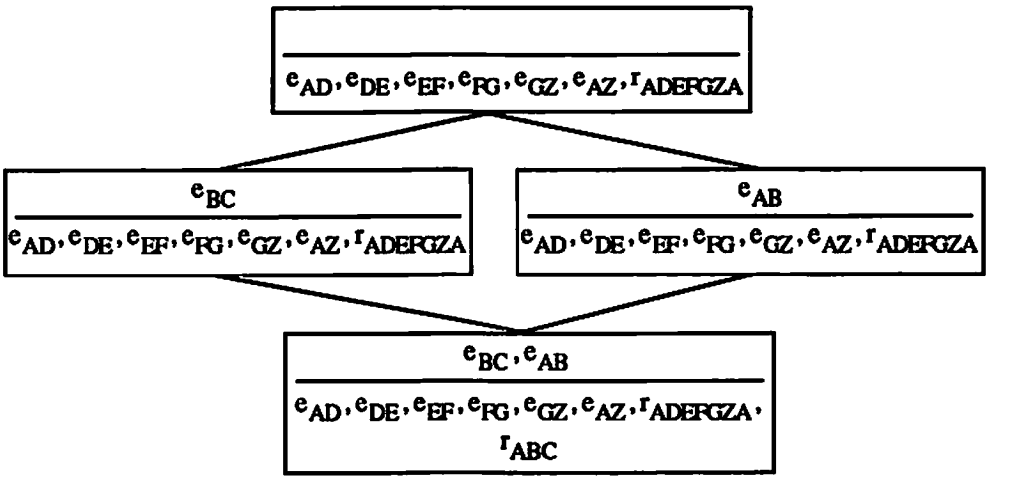


Figure 4.21: Effect of confirming the roof hypothesis

- To ensure that the line segment is on the shadow side of the edge, first determine the point p_S that is a distance Δ_λ away along $\vec{L}$ from the midpoint of the edge e_{AB}. Then test to see that the points p_S and p_R are on the same side of the edge e_{AB}. If the coordinates of these points are (x_S, y_S) and (x_R, y_R) then this constraint can be quickly tested for, using the *parameter* slots of the edge e_{AB}.

  $(\omega_{x-e}x_S + \omega_{y-e}y_S + \omega_{0-e}) \times (\omega_{x-e}x_R + \omega_{y-e}y_R + \omega_{0-e}) > 0$

- The corresponding shadow length is then estimated as an average:
  $\frac{d_Q+d_R}{2} \times \frac{|\vec{L}|\times|\vec{n}|}{\vec{L}\cdot\vec{n}}$
  A list of the line segment name and this length is then stored in the shadow-line slot of the edge.

Another rule then examines each shadow casting edge, retains the nearest such correspondence and deletes the rest. To confirm this correspondence, yet another rule examines the region between the edge and the line segment as shown in Figure 4.20(b) by sampling the hypothesized shadow region. The sampling

interval is specified by the user. If the mean of these samples is less than a certain threshold and the standard deviation is within the uncertainty in gray level Δ_g, then the correspondence is confirmed. Else, the correspondence is deleted. This is an expensive test and so was delayed till at most one correspondence was retained and all the others deleted. Since a closed-ring will have more than one shadow casting edge these could lead to different shadow interpretations for different shadow-casting edges. The correspondence with the maximum height interpretation is chosen. This is because some of the shadows fall on adjacent buildings and lead to a reduced height interpretation. A detailed analysis of shadows cast by polyhedra and the constraints they impose on surface orientations is given in [105, 106]. However, no noise is assumed in this theory and the issue of efficient computation is not addressed. Our work involves a restricted class of polyhedra. However, we account for noise and for the different constraints in a computationally efficient way. Shadow analysis has been used for interpretation of buildings in a few aerial image interpretation systems [64, 66, 107]. Lowe [108] developed a program that recovers the 3D description of shadow casting object boundaries from their correspondence with shadow boundaries. However, the boundaries were hand generated and free from errors.

Consider a closed-ring, all of whose edges have adequate low level support. If it has a height interpretation through shadow correspondences then that leads to a roof hypothesis. In the current strategy, this interpretation is automatically *confirmed*, leading to automatic restructuring of the context tree and a reduction in the number of contexts. The context restructuring is taken care of by the TMS. The problem solver just needs to communicate to the TMS that a particular context is to be *confirmed.* This boundary between the problem solver and

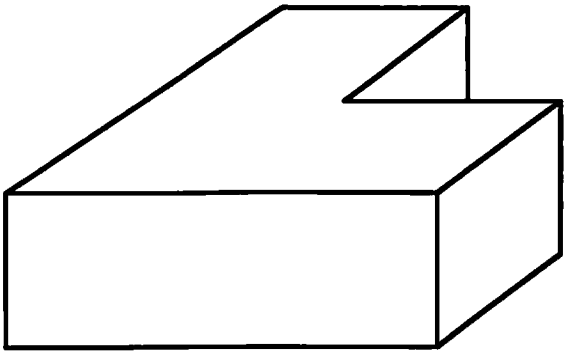


Figure 4.22: Wire frame interpretation

the TMS is natural and the problem solver need not worry about the book-keeping rituals, which are automatically handled by the TMS. This higher level of abstraction has the salutary effect of making the system very understandable and so easy to revise. In Figure 4.19, all the edges of the closed-ring $r_{ADEFGZA}$ are found to have adequate support and a height interpretation is determined from the vertex correspondences. The effect of *confirming* the edge-ring is shown in Figure 4.21. A surface based description of a building is constructed for each confirmed roof interpretation. The walls of the building are assumed to be perpendicular to the ground plane. This is illustrated in Figure 4.22.

4.2.6 Multiple interpretations

For an illustration of how competing contexts can be generated, consider Figure 4.23. Figure 4.23(a) shows a simple case where there are 8 roof edge assumptions e_A, e_B, e_C, e_D, e_E, e_F and e_G. Since e_D and e_E (as well as, e_C and e_F) intersect, the merge of their contexts is marked as a *nogood* [4] by the TMS and no context that has an environment that is a superset of the *nogood* environment is ever generated by it. This effectively prunes the search space. Figure 4.23(b)

shows one possible scenario during the edge-ring composition process. The *environment* and the deductions based on it are shown separated by a horizontal black line. Suppose at this stage, all the edges of the edge-ring r_{ABCD} are found to have sufficient low level support and the edge-ring has shadow confirmation, then the context containing this edge-ring is confirmed by the system. This collapses the part of the context tree that forms this context into the root context. This reduces the size of the context tree and also makes this information available to all other contexts by virtue of context inheritance. This restructuring of the context tree is automatically handled by the TMS. The result of such an action is shown in Figure 4.23(c). The roof edge assumptions e_E and e_F are eliminated from the database by the TMS. Suppose instead that r_{ABCD} is found to have insufficient low-level support, then the situation shown in Figure 4.23(d) could result. The database is effectively split into conflicting interpretations r_{ABCD} and r_{EFGH}. In the current strategy, when two edge-rings are in competing contexts, the context of the one with the smaller low-level support is marked as contradictory. When the context generation process has reached a quiescent state (i.e. no more contexts are being generated) all the surviving roof interpretations can be confirmed.

The advantages of using a multiple-context search strategy with the aid of an ATMS over ordinary single-context search schemes (like depth-first, breadth-first, best-first etc.) have been well documented in [4]. We have attempted to make a list of how it specifically assisted us in the building detection process.

rediscovery of inferences No edge-ring is ever rediscovered in the composition process. The TMS ensures that no environment is reproduced in the

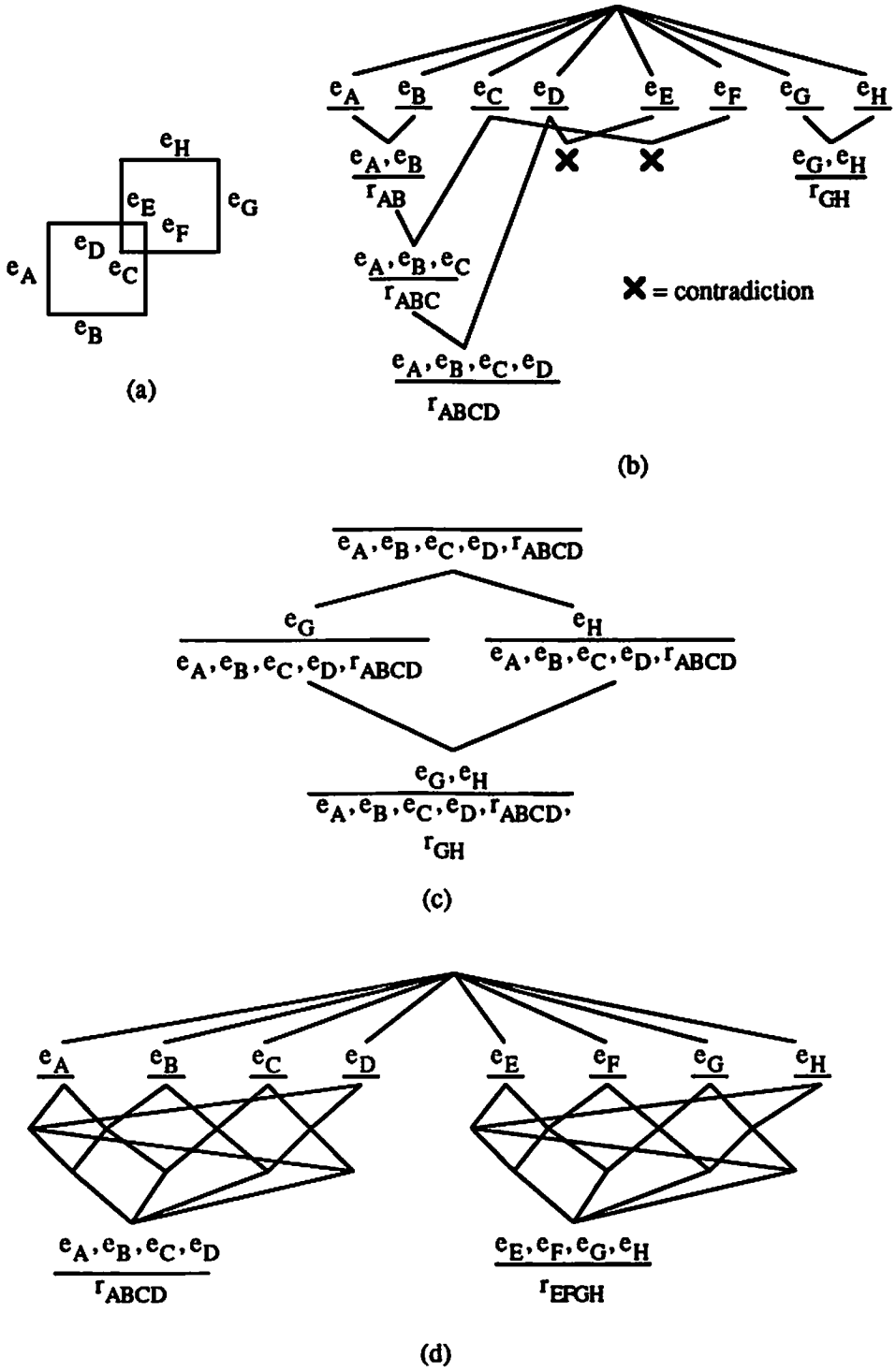


Figure 4.23: (a) 8 roof edge assumptions. (b) Edge ring composition. (c) Effect of confirming the context with r_{ABCD}. (d) Competing contexts.

context tree. Contexts which have environments that are supersets of this environment, automatically inherit all the inferences. Since, no environment is ever recreated and inheritance permits the sharing of information, there is virtually no rediscovery of information when using the ATMS. There is no such guarantee when using a simple search scheme.

rediscovery of contradictions If an edge-ring is in error (for example, self intersecting), then the environment of that context is stored as a *nogood.* The TMS never creates any context that has an environment that is a superset of the nogood environment. Consequently, there is no rediscovery of contradictions. A simple search procedure could rediscover this contradiction.

transfer of information through the search space If a roof hypothesis is confirmed because of overwhelming low level evidence, then all the underlying object edge assumptions are correct. This information must be transferred to other points of the search space. The ATMS does this automatically by collapsing the confirmed context into the root context. The information is then available in all other contexts through inheritance. There is no such simple way to transmit information in simple, single context search schemes. This process was illustrated previously in Figure 4.21.

differential diagnosis In Figure 4.30, the highlighted circular area shows three mutually competing interpretations for a roof. After comparing the underlying low level support, one interpretation was retained and the competing ones *contradicted.*

incremental nature of search During the edge-ring composition process new edge assumptions are added top down and they incrementally participate in the search. In an ordinary search scheme, the search cannot deal with new assumptions, unless the program is rerun.

opportunistic search We used a *best-contexts-first* search strategy that is completely opportunistic. This would not have been possible with standard search methods.

modifiability We iterated through several search strategies before arriving at the current one. Since, ATMSs are typically designed around rule-based systems, the search strategy is explicitly specified in modular units of production rules. This allows for experimentation to choose a good strategy.

4.2.7 Real image examples

A typical aerial image (IMAGE-A) is shown in Figure 4.24. The image is 320 pixels wide and 320 pixels long. The angular uncertainty Δ_θ was modeled as 6.0°, the positional uncertainty Δ_λ as 1.9 pixels and the gray level uncertainty Δ_g as 12. The line segments detected in this image are shown in Figure 4.25. There are 836 line segments in this figure. The vertices and limbs detected from these line segments are shown in Figure 4.26. The black circles are the vertices. The arrows pointing away from the vertices are *limbs*. There are 266 vertices and 560 limbs. Most vertices have two limbs and some have three. In Figure 4.27, the hatched lines (along the direction of the projection of the illumination vector) connect vertices in shadow correspondence. There are 32 correspondences. The initial set of edges hypothesized from Figure 4.26 by looking for coincident

limbs are shown in Figure 4.28. There are 454 of these edges. However, those edges with insufficient low level support are deleted and the number of edges is reduced to 184 in Figure 4.29. In Figure 4.30, the 41 new edges hypothesized by the use of free-ring completion heuristics on the data in Figure 4.29 are shown along with the original edges. Figure 4.31 shows the edge-line shadow correspondences for the shadow casting edges of some closed-rings. There are 13 such correspondences. A surface based description of a building is constructed for each confirmed roof interpretation. The walls of the building are assumed to be perpendicular to the ground plane. The entire system was implemented in LISP and ART [5] (Automated Reasoning Tool) on a Texas Instruments Explorer LISP machine and took approximately 14 minutes of user time to go from the line segments to the buildings. A wireframe rendering of the final interpretation is shown in Figure 4.32. In this figure only the faces visible from the current viewpoint are shown. However, there are no corrections for occlusions. Occlusions are taken care of in the gray scale rendering in Figure 4.33. A ray casting algorithm is used. The original image is used as a texture map to paint the roofs of the buildings. The walls of the buildings are rendered using a lambertian reflectance model. The rest of the image is painted black.

For another real image example, consider the aerial image (IMAGE-B) shown in Figure 4.34. The image is 352 pixel wide and 288 pixels long. Δ_θ was specified as 6.0°, Δ_λ as 2.2 pixels and Δ_ρ as 12. The 96 line segments detected in this image are shown in Figure 4.35. The 34 vertices and 69 limbs detected from the line segments are shown in Figure 4.36. There is one shadow vertex correspondence as shown in Figure 4.37. Initially 33 edges were created as shown in Figure 4.38. These were pruned to 29 in Figure 4.39. There are no free-ring completions in this

image. Figure 4.40 shows the 3 edge-line shadow correspondences. Figure 4.41 shows the wire-frame interpretation and Figure 4.42 shows the rendering. The system took about 4 minutes of user time to go from the line segments to the buildings.

4.3 Discussion

This chapter outlines a building detection system that we developed. Shadow evidence is used to estimate the heights of buildings. The system makes some simple assumptions. First, buildings are modeled as roofs with orthogonal corners and walls as vertical faces. Then, we assume that shadows fall on level ground and are clearly visible. The sun angle is assumed to be known. These assumptions may seem restrictive but there seems to be no other alternative when dealing with monocular images. If a stereo pair of images are available then these assumptions can be relaxed. The next chapter reviews a stereo system that can be useful in this task.

Figure 4.24: A typical aerial image – IMAGE-A.

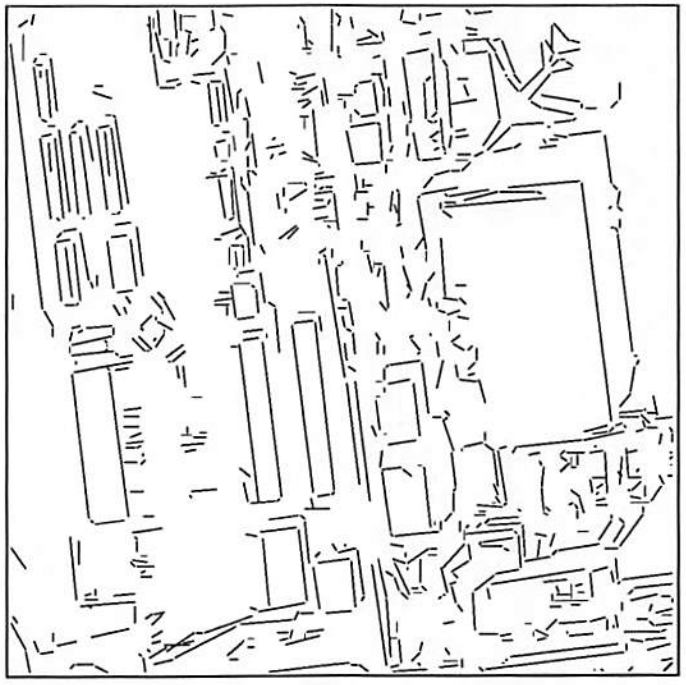


Figure 4.25: Lines extracted from the image.

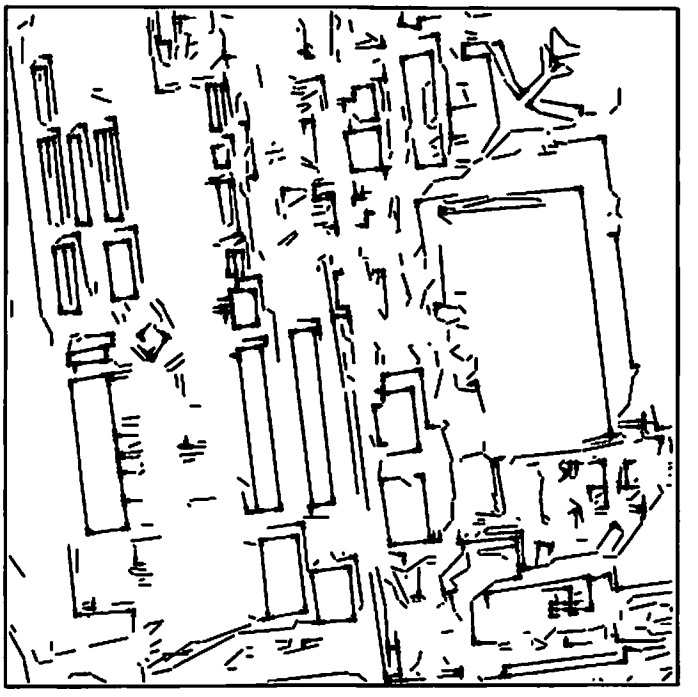


Figure 4.26: Vertices and Limbs.

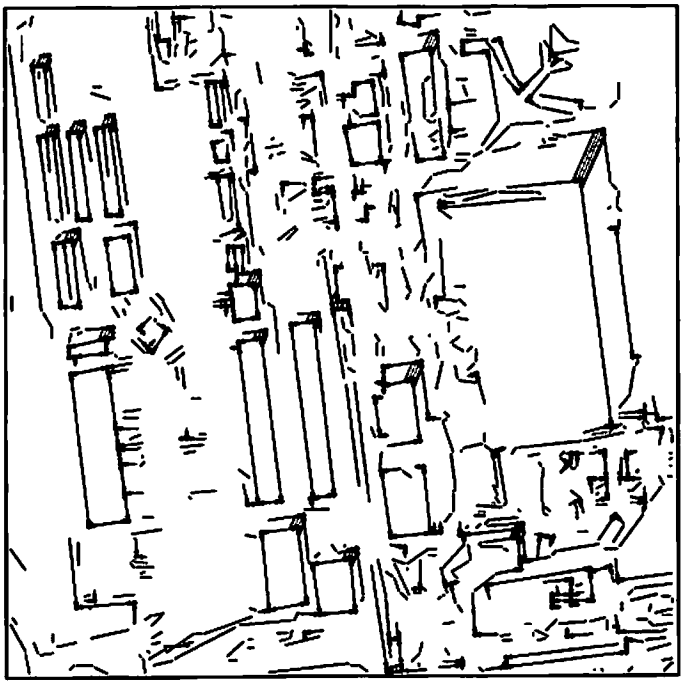


Figure 4.27: Vertices in shadow correspondence.

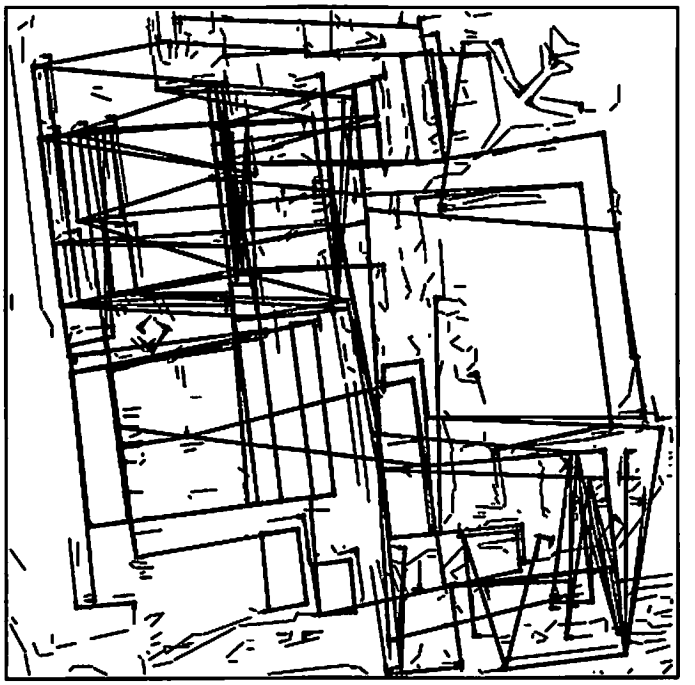


Figure 4.28: Initial set of edges generated.

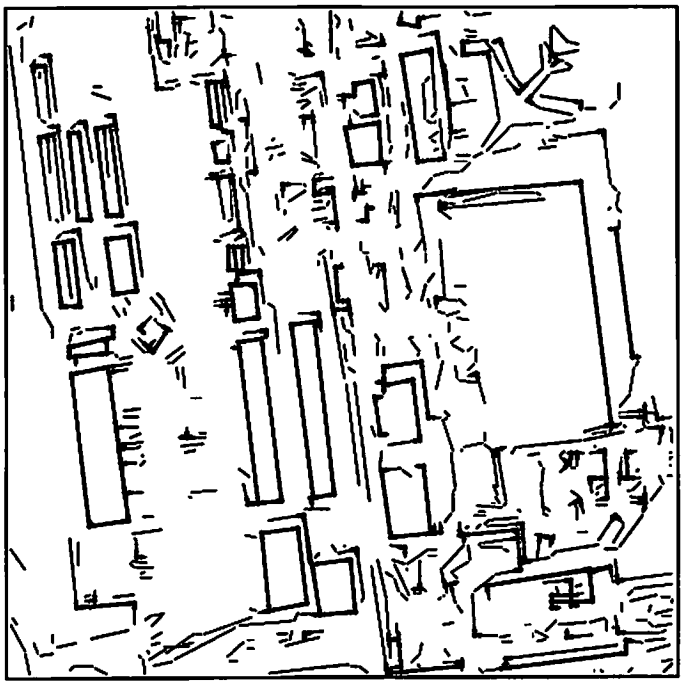


Figure 4.29: Set of edges after deleting edges with insufficient low level evidence.

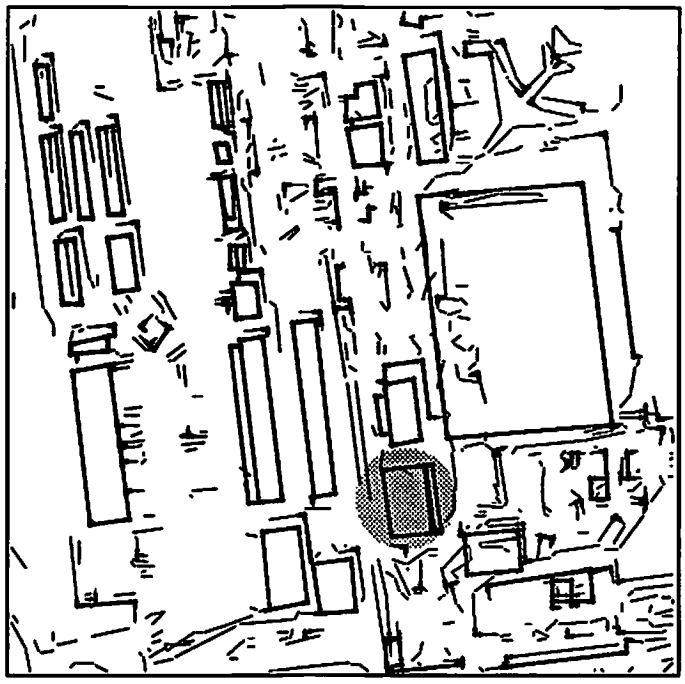


Figure 4.30: Edges created bottom-up and top-down.

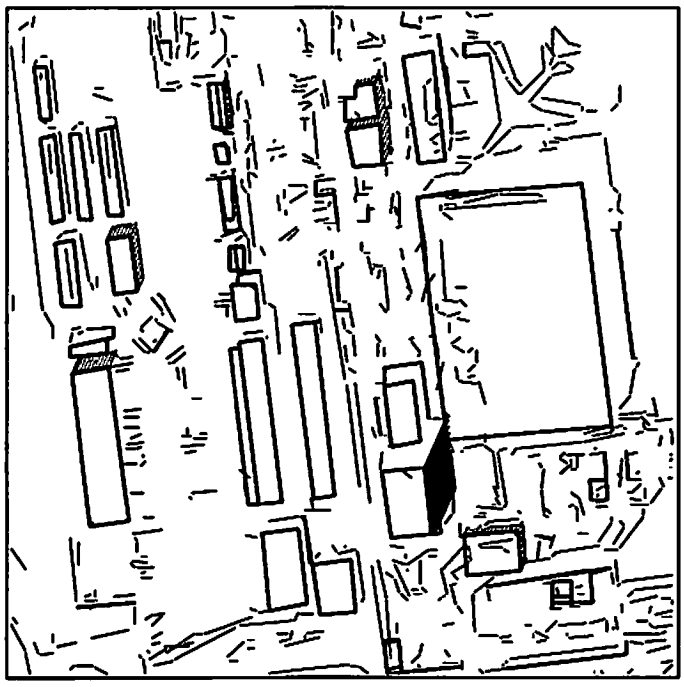


Figure 4.31: Edge-line shadow correspondence.

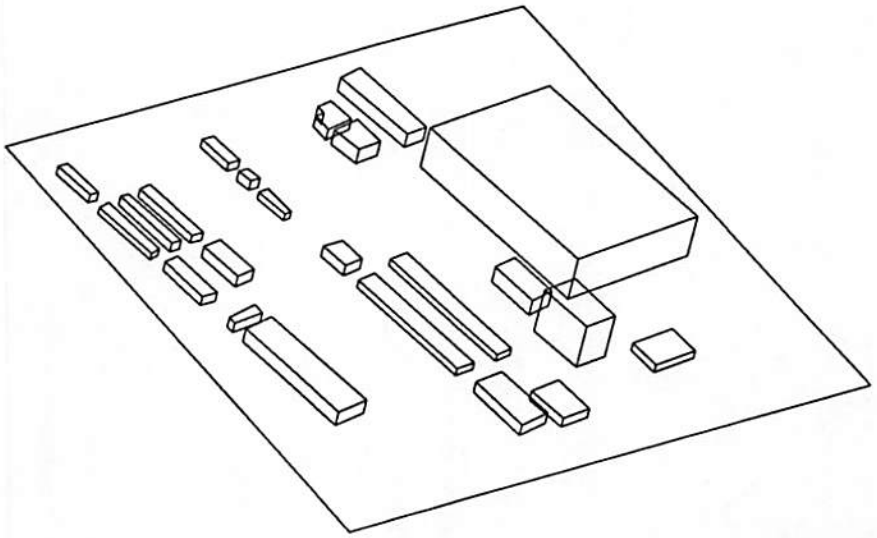


Figure 4.32: The final wireframe interpretation.

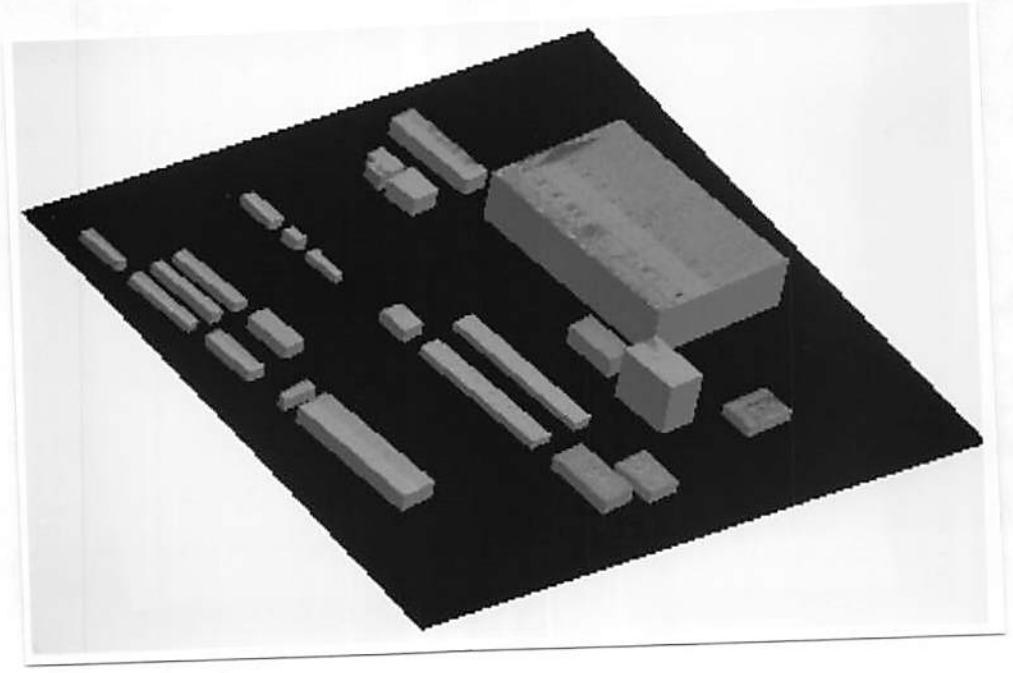


Figure 4.33: Rendering of the interpretation.

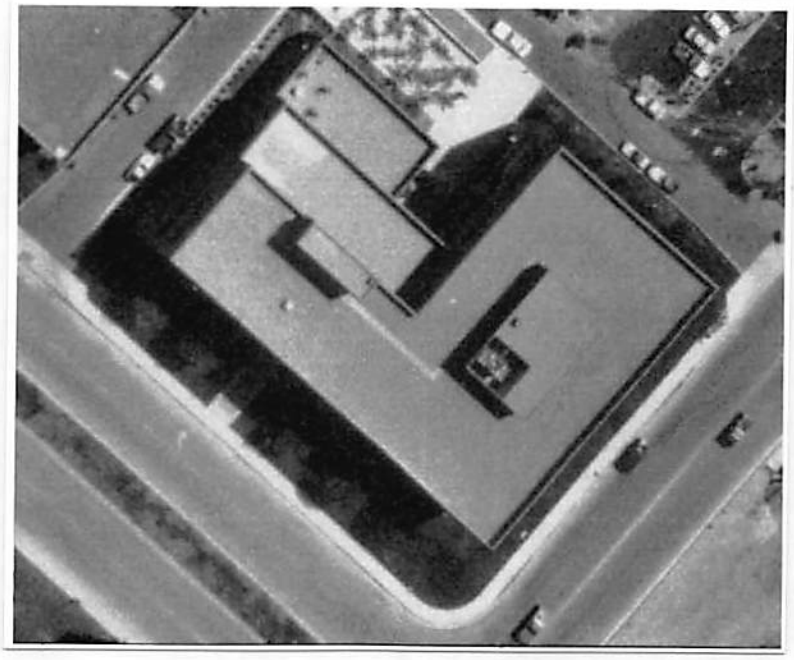


Figure 4.34: A typical aerial image – IMAGE-B.

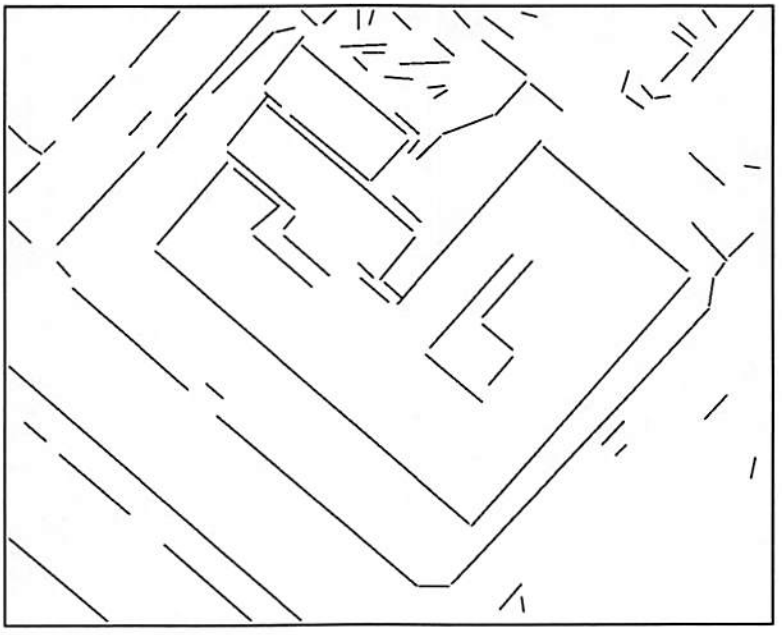


Figure 4.35: Lines extracted from the image.

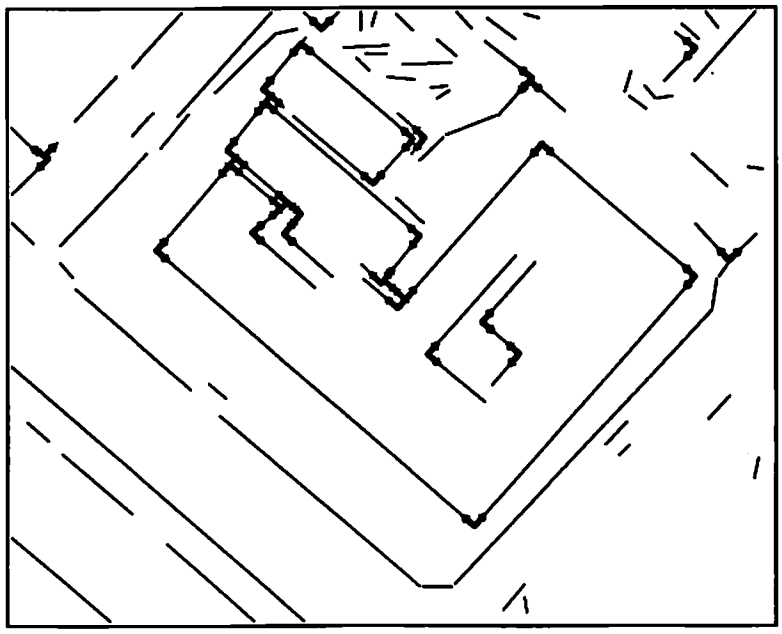


Figure 4.36: Vertices and Limbs.

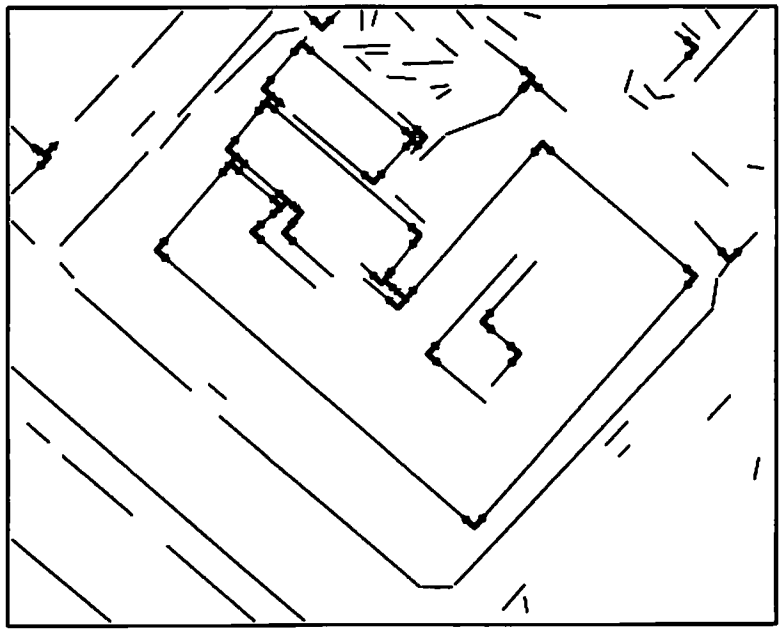


Figure 4.37: Vertices in shadow correspondence.

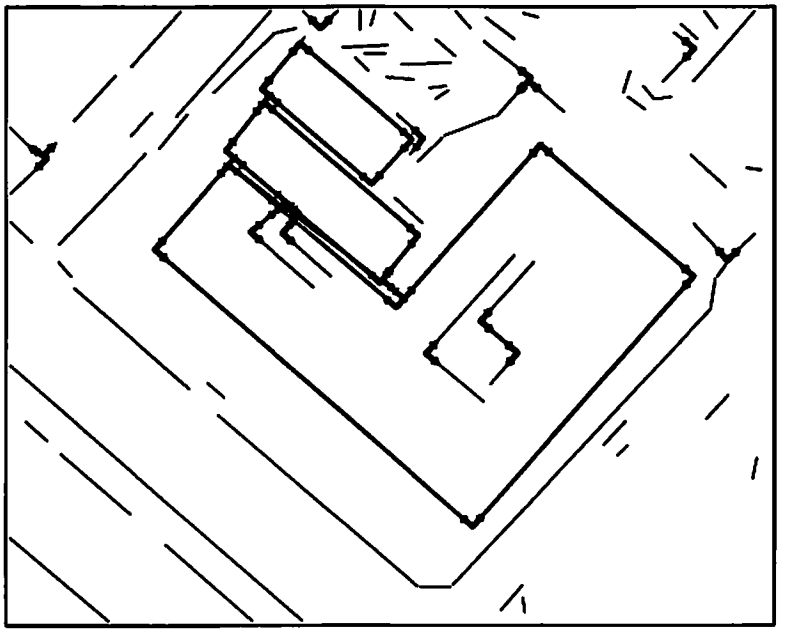


Figure 4.38: Initial set of edges generated.

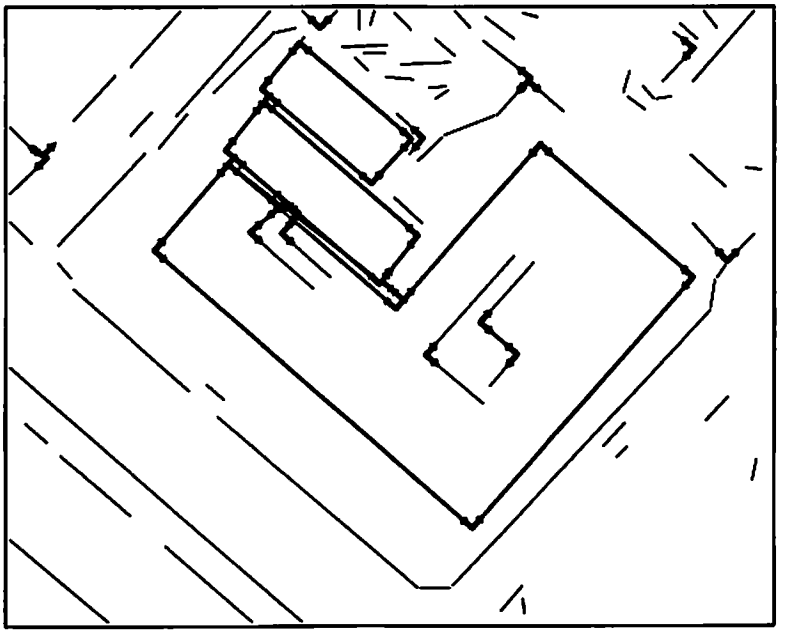


Figure 4.39: Set of edges after deleting edges with insufficient low level evidence.

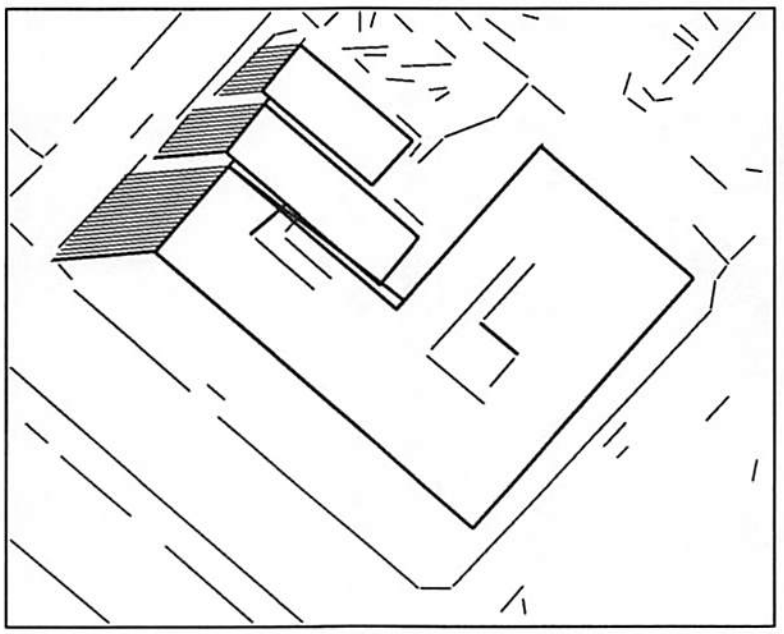


Figure 4.40: Edge-line shadow correspondence.

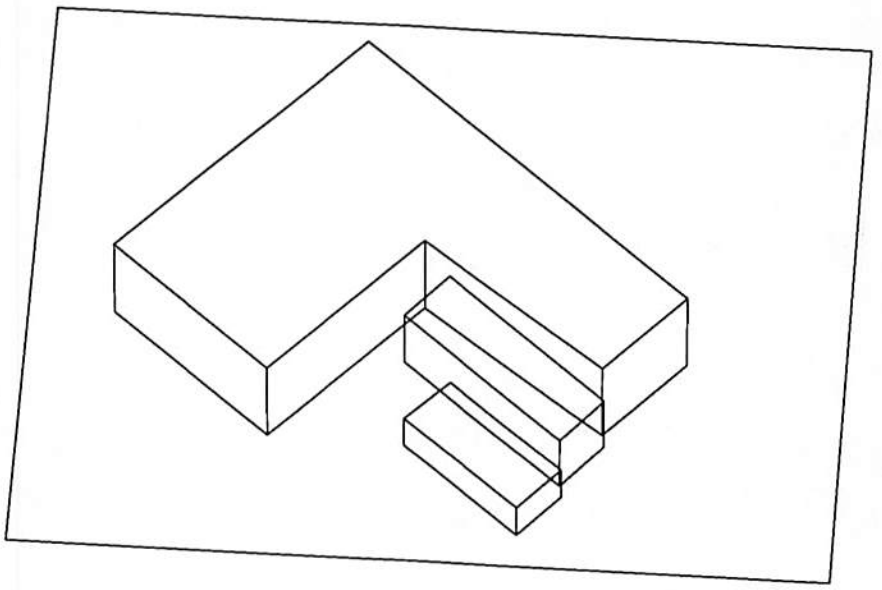


Figure 4.41: The final wireframe interpretation.

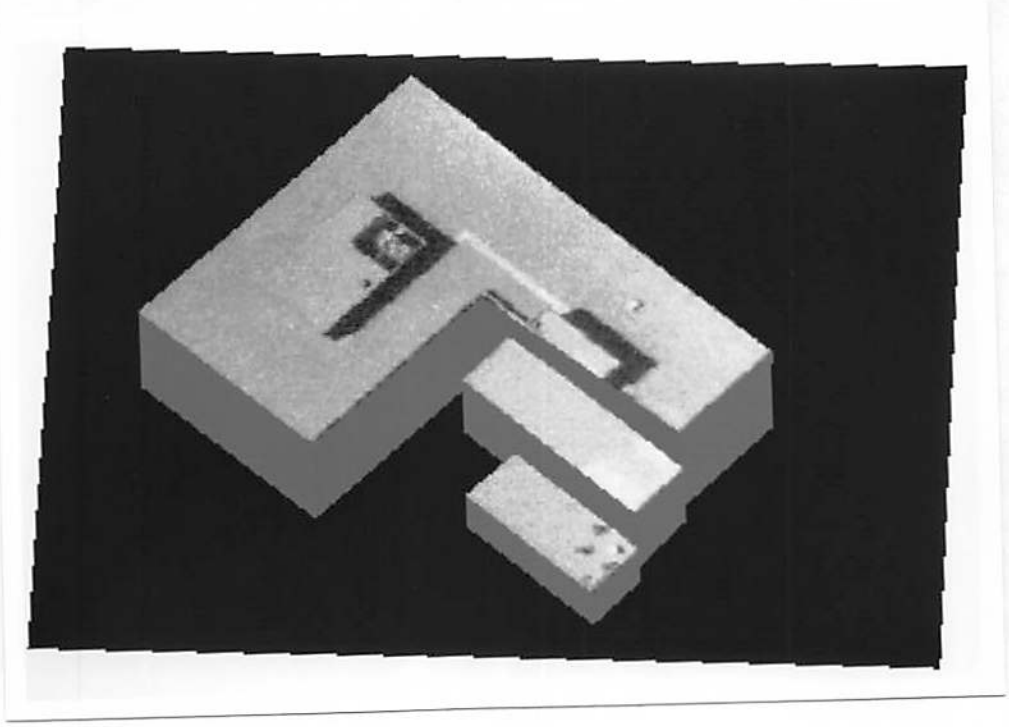


Figure 4.42: Rendering of the interpretation.

Chapter 5

Hierarchical Stereo and Motion Correspondence

5.1 Introduction

In this chapter we first focus on stereo matching. Motion correspondence is then described as an extension to the stereo matching algorithm. The basic idea in our stereo matching approach is to extract as much structure as possible from each scene before matching. This includes both topological and perceptual structure. This is accomplished by a hierarchical grouping process that group line segments into complex structures. Relations between the structures are also computed. The hierarchy consists of *lines*, *vertices*, *surface edges* (edges, for short), and *surfaces*. The relations include *parallel*, *collinear*, *left-of* and *right-of*. This is similar to the feature hierarchy used in building detection, but there is no restriction that vertices must have lines that are orthogonal to each other. The objective is to match the hierarchical relational graphs extracted in both scenes.

Matching starts at the highest level (surfaces) and proceeds to the lowest (lines). At each level matches are grouped into strong clusters based on *structural* and *perceptual* relations. These groupings are then confirmed.

Two kinds of ambiguity can be identified in a hierarchical matching system: ambiguity involved in building the feature hierarchy (how close must the end points of two lines be, before they can be grouped into a vertex?) and ambiguity in the matches (is a match purely accidental or is it correct?). To proceed in face of this uncertainty, we treat the features and their matches as hypotheses whose beliefs can later be revised (to true or false) as more evidence accumulates. When these hypotheses are grouped, they form contexts. Constraints are specified by the user as *nogoods*. These constraints identify certain contexts as contradictory. A TMS manages this search space made up of hypotheses, their groupings (contexts) and constraints. The TMS ensures that the final solution contains no contradictions.

Section 5.2 describes the feature hierarchy and its representation. Section 5.3 describes the hierarchical matching process. Section 5.4 discusses how match groupings are used to form islands of certainty. Grouping constraints and the use of a TMS to implement these constraints are discussed in Section 5.5. Section 5.6 shows some experimental results. Section 5.7 discusses the extension of this matching technique to motion correspondence. Section 5.8 presents results on tracking point and line features over several frames of an image sequence.

5.2 A Feature Hierarchy

The proposed hierarchy, shown in Figure 5.1, consists of lines, vertices, edges, and surfaces (contiguous sets of edges, i.e. edge-rings). Surfaces can be open or closed. Currently objects are not included in our hierarchy, mainly because in most of the images that we used, surfaces were the most complex structures visible; but the extension is straightforward. The domain of applicability is any scene whose contours can be approximated with a series of straight line segments (they need not lie on a plane). This includes urban, indoor, and factory scenes.

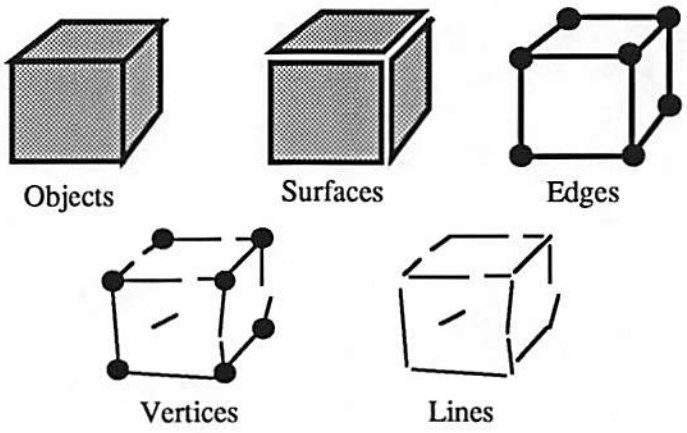
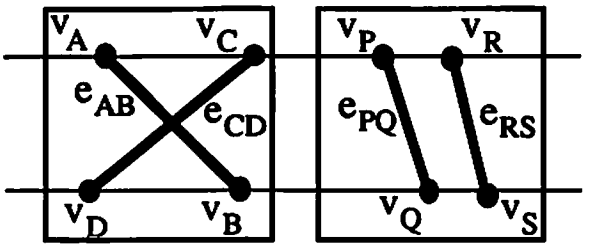


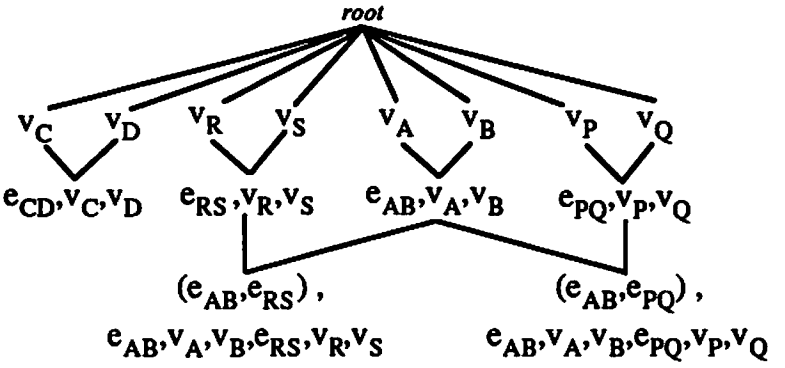
Figure 5.1: A feature hierarchy

The initial line data given to the system is noisy. The end points are not well localized and some lines are fragmented into segments. This is a direct result of the noisy nature of real images and the fact that low level techniques, including edge pixel detection and line extraction, are imperfect. This leads to uncertainties when grouping lines into higher level features. To deal with these uncertainties, features in the hierarchy (other than lines) are conjectured as *hypotheses*. These features may be *confirmed* when more evidence becomes available, including evidence of a matching feature in the other image. The

hypotheses and their groupings define a context graph (a directed acyclic graph). This is illustrated in Figure 5.2 which shows the context graph for a simple stereo pair. The context graph is the search space of the problem. Nodes in the graph correspond to collections of hypotheses. The objective is to incrementally build this graph and identify those nodes that correspond to partial solutions.



(a) An example stereo pair.



(b) Context graph

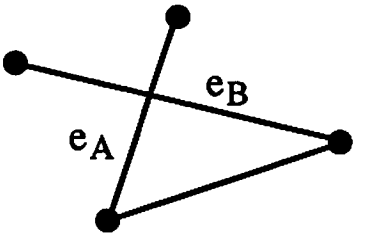
Figure 5.2: Hypotheses and their groupings form a context graph. v_A, v_B, v_C, v_D, v_P, v_Q, v_R, and v_S are vertex hypotheses. e_{AB}, e_{CD}, e_{PQ}, and e_{RS} are edge hypotheses. (e_{AB}, e_{PQ}), and (e_{AB}, e_{RS}) are edge-match hypotheses.

We now describe how the feature grouping process is implemented in each scene. The strategy is essentially the same as in building detection. The difference is that vertices, edges, and surfaces are all treated as hypotheses. Also perceptual relations between elements at the same level of the hierarchy are

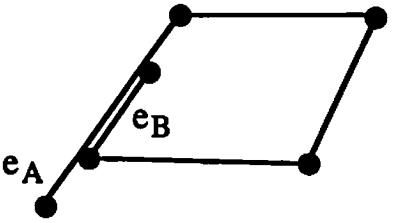
determined. These include parallel, collinear, proximate (left-of or right-of) relations. As usual, bucketing techniques [46, 109] are used to reduce the number of comparisons necessary to compute these relations. Two lines whose end points are close to each other are possibly connected in the scene. A *vertex* is hypothesized to join the two lines in an L-junction. The position of the vertex is at the intersection that is obtained if the lines were extended. *Edges* are sets of collinear line segments with vertex terminations. They represent surface boundaries in the scene that are fragmented into lines (because of poor contrast, etc). For extracting edges, if two vertices in the scene have lines that are collinear and face each other, an edge is hypothesized to join the vertices. This procedure generates several spurious edges, because of accidental alignment of vertices. To eliminate such edges, all lines that overlap each edge are accumulated. Then the percentage overlap of each edge with these lines is calculated. This is the same approach used in the building detection system. This gives the line segment support for each edge. All the edges with insufficient line segment support (less than 90%) are deleted. A few spurious edges may still remain. However, it is unlikely that such edges will have a corresponding match in the other scene.

Our strategy for finding connected set of edges to identify surfaces is a little different from that used in building detection. In this case, there is no top-down search for edges to close open rings. Also, there is no orthogonality restriction when finding connected edges. Any two edges that share a vertex are linked through a *connects* relation. A surface (edge-ring) is an ordered set of *connected* edges (at least two) that cannot be extended any further. Each edge in an edge-ring is connected to the next. Further, we expect an edge-ring to correspond to a surface contour in the scene. So they cannot assume unnatural

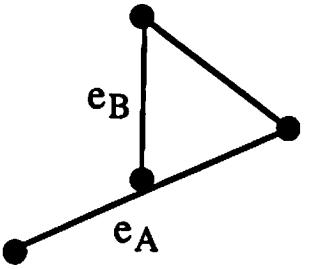
forms. (Figure 5.3). To preclude the formation of such edge-rings we explicitly store incompatibility relations. These include *intersects*, *overlaps* and *touches* relations. The *intersects* relation links intersecting edges. The *overlaps* relation links overlapping edges. The *touches* relation links two edges which touch each other but do not *intersect*, *overlap* or *connect*. Such inconsistent groupings of edge hypotheses are identified with nogoods (see Section 5.5).



(a) Intersecting edges.



(b) Overlapping edges.



(c) Touching edges.

Figure 5.3: Inconsistent edge-rings.

As described earlier, the features in the hierarchy are all hypotheses (assumptions). For the extraction of edge-rings, consider the subgraph in this feature hierarchy whose nodes are the edge hypotheses, whose compatibility links are the *connects* relations and incompatibility links are the *intersects*, *overlaps* and *touches* relations. Edge-rings are the paths in this graph (an ordered set of *connected* nodes, where each node is connected to the next). The paths must be

maximal, i.e. no further nodes can be added at the ends. These edge-rings are formed by merging connected edge hypotheses together. Inconsistent groupings are eliminated by the TMS. Closed-rings in the graph, which have a high probability of corresponding to real surface contours, are confirmed. Then all their underlying feature hypotheses are also confirmed. The effect is that these feature hypotheses are now truths and are asserted into the root context. They are then inherited in all contexts. Those contexts with edge hypotheses that *intersect*, *overlap* or *touch* these confirmed edge hypotheses are eliminated (negated) from the database by the TMS (based on nogoods). For example, consider the closed-ring *abcd* in Figure 5.4. The effect of confirming *abcd* has its ramifications. Edge hypotheses *a*,*b*, *c*, and *d* are now true features. Edge hypothesis *e* is negated as it violates *d* (intersection). Now $hgfe$ is no longer a valid edge-ring, but hgf is.

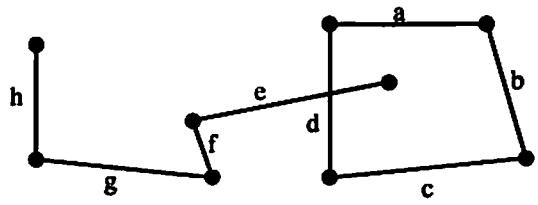


Figure 5.4: Confirming a closed ring.

A related issue is how to represent the hierarchy in a computer for computational purposes. The language for representation must be concise and natural, have a strong expressive power and must be easy to access [2]. We have found that no one representation language meets all these criteria [110]. So we chose to use a dual representation of the feature hierarchy and the relations between the features. One is a semantic network or relational graph representation [111]. Nodes in the graph correspond to features (lines, vertices, etc.) with associated

feature descriptors (such as end point coordinates of lines, etc). Arcs are relationships (parallel, collinear, left-of, right-of, etc). This network is also translated into a set of predicate calculus type propositions (Figure 5.5). Each proposition is a triplet. The first element of the triplet is a relation and the other two are objects connected by this relation. Any additions or deletions to the database in one representation are immediately reflected in the other. The two representations are identical in information content, but differ in information organization. It may seem wasteful to duplicate information, but each representation has its advantages and the increase in memory is more than offset by the gain in system flexibility and power. The semantic net representation is well suited for indexing [111]. For example, the lines that compose a vertex are easily retrieved by just following the vertex-has-line links of the vertex. Similarly, vertex coordinates are easily obtained from vertex node feature descriptors. The propositional representation is ideal for general inference mechanisms, like production systems [112] and theorem provers. In our system, the feature grouping and matching strategy is encoded in production rules. The propositional representation is also useful in the explicit representation of constraints as nogoods (Section 5.5).

5.3 Matching the Hierarchy

The feature hierarchy together with feature relationships forms a relational graph (or a semantic network). The objective is to match the graphs derived from a stereo pair. We proceed by finding matches for each feature and grouping these matches into local cliques. We start by matching at the surface level and proceed to the line level. For two features to match, their component features

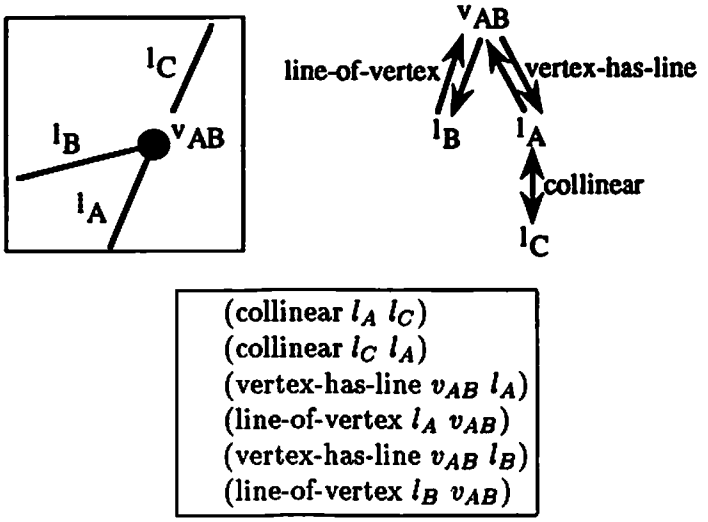


Figure 5.5: A simple picture, its semantic net, and propositional representations.

must match, recursively. For example; two surfaces match only if their edges match (in the same order), their edges match only if their vertices match, and so on. At each level those features that were not matched at previous levels (as component features) participate in the matching process. Below we detail the unary constraints used before hypothesizing a match between features.

epipolar constraint: Consider a point in the left (right) image. Its corresponding point in the right (left) image lies on an epipolar line defined by the camera geometry [75]. For each feature in one image, we search for a matching feature in the other image, within the epipolar lines spanning the feature. To account for uncertainty in the positions of the features, we tolerate a deviation of about 4 pixels perpendicular to the epipolar lines. We consider a camera geometry where the two cameras are laterally displaced with respect to each other. This results in epipolar lines that are coincident with the scan lines of the image. In case the camera geometry

is different, the images can be rectified so that scan lines correspond to epipolar lines [113]. Disparity can then occur only in the horizontal direction. If disparity bounds are specified, then matching is further restricted to a portion of the epipolar line. In the case of lines, the search area for matching is an epipolar window in the other image. This is shown as a shaded region in Figure 5.6). The size of the window is determined by the line and the allowed disparity range. Any line that passes through this window is a potential match. The epipolar window is necessitated by the fragmented nature of lines. Edges, are however not fragmented and the search for matches can be restricted to epipolar lines of their end points. Any edge in the other image that has one of its end points on each of these epipolar lines (within the allowed disparity and within 4 pixels from the epipolar lines) is a potential match.

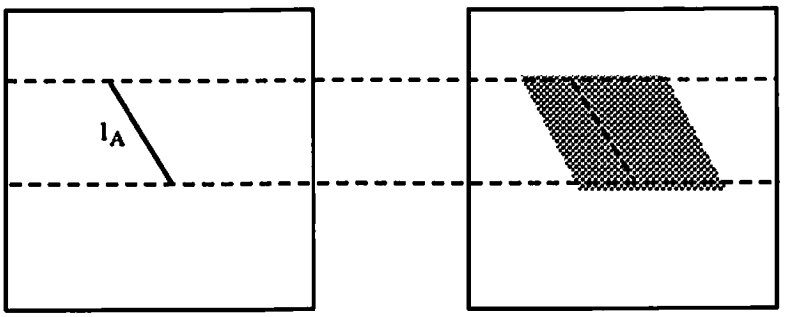


Figure 5.6: epipolar constraint.

area and length constraints: We expect the areas of matching surfaces to be within 75% of each other. The lengths of matching edges must be within 75% of each other and those of lines must be within 25% (to account for their fragmentation).

orientation constraint: Since the two views in a stereo pair are not very far apart we expect the orientations of features to be similar [114]. So we bound the orientation difference between matching features at 30°. Similar bounds have been used in other work [45, 47]. For two vertices that satisfy the epipolar constraint, the orientation constraint applies to their matching lines. Further, the matched lines must be in the same relative order (clockwise or anti-clockwise).

contrast constraint: For two linear features (lines or edges) to match, their contrast must be similar. Each line or edge marks a boundary between a dark and a bright region. If the bright regions are on the same sides of the linear features (to the left or right) then they can be matched.

5.4 Match Groupings

For each feature in one image there may be several match hypotheses in spite of the unary constraints detailed in the previous section. A more global context is needed to disambiguate between competing matches. Ambiguity is least at the surface level where features have the most structure and is greatest at the line level, where typically a line has several possible matches. So we start by first matching at the surface level. To reduce ambiguity even further, we group compatible matches into local cliques. Compatibility is based on consistent structural and perceptual relations across the scenes (Figure 5.7). Structural groupings are based on unmatched features at the previous level of the hierarchy. These unmatched features are used as foci-of-attention when matching. For example, consider the groupings when matching at line level.

Figure 5.7 shows a typical case. Vertex v_{AB} has no match in the other image. But its lines l_A and l_B match l_P and l_Q respectively. Further, these lines when extended intersect on the epipolar line of v_{AB}. This is consistent with the presence of a vertex at this location. This vertex was not detected earlier during the feature grouping process because the lines l_P and l_Q were not closer than a threshold. Unmatched vertices serve as foci-of-attention when matching at the line level. The system focuses on these vertices and creates structural groupings. The matches (l_A, l_P) and (l_B, l_Q) together with vertex v_{AB} form a structural grouping. Similarly, match hypotheses (e_A,e_P) and (e_B, e_Q) form a structural grouping because the edges are parts of unmatched surfaces in both scenes. Perceptual groupings are based on consistent perceptual relations [17] across the scenes (in case of surfaces and vertices, perceptual relations are based on relations of their constituent edges and lines respectively). These include *parallel*, *collinear* and *proximate* relations. In Figure 5.7, l_C and l_D as well as their matches l_R and l_S are connected through the *parallel* relation. So the match pairs (l_C, l_R) and (l_D, l_S) form a parallel perceptual grouping.

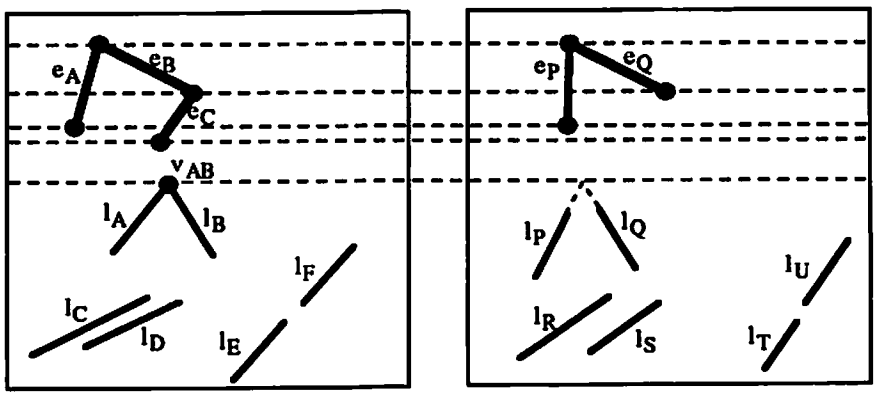


Figure 5.7: Match Groupings.

Those groupings that share a match are formed into super-groups. Groupings are currently restricted to a size of three consistent matches. These groupings correspond to partial solutions to the matching problem. So our search graph is only three levels deep. This reduces matching complexity. We have found that these groupings (together with the hierarchical feature grouping process) are usually sufficient to completely disambiguate between competing matches, i.e. it is hard to find consistent groupings for an erroneous match hypothesis. The groupings form islands of certainty as in blackboard systems [115, 116]. At each level structural groupings are confirmed first, followed by perceptual groupings. Among perceptual groupings, proximate groupings are confirmed last. Proximate groupings are those matches that have features that are *proximate* in both scenes and have a similar disparity. However, the constraint that adjacent features in the scene should have similar disparity is a weak constraint, so these groupings get the lowest priority. Other researchers have chosen to use the similar disparity criterion as their primary constraint [45, 46]. Larger groupings are confirmed before smaller groupings (groups of three first, followed by groups of two and then ungrouped matches) because they have a more global context. Within each size the confirmation is random.

When a grouping of match hypotheses is *confirmed* the beliefs of several hypotheses are affected and must be revised. The process of *belief revision* is handled by the TMS. First, all underlying hypotheses (both feature and match) of the confirmed grouping are converted to truths, by collapsing the portion of the context graph that supports the grouping into the root context. These root hypotheses then are made visible in all contexts where they are grouped with the other hypotheses. This may lead to some contradictory groupings. The contexts

corresponding to such groupings are eliminated by the TMS. The strategy of confirming partial solutions and eliminating contradictory contexts has the effect of preventing unbridled growth in the size of the context graph. For example; consider the context graph shown in Figure 5.2. Suppose the match hypothesis (single grouping) (e_{AB}, e_{PQ}) is confirmed. Then, the following actions take place:

1. The portion of the context graph supporting this grouping is collapsed into the root. The result is shown in Figure 5.8(a). The root hypotheses v_A, v_B, v_P, v_Q, e_{AB}, e_{PQ}, and (e_{AB}, e_{PQ}) are now visible in all contexts by inheritance (but for clarity are not shown explicitly in the non-root contexts of the figure).

2. This creates two contradictory contexts. The context with match hypothesis (e_{AB}, e_{RS}) is invalid because it violates the match (e_{AB}, e_{PQ}) (uniqueness constraint, see Section 5.5). Similarly, the context with edge hypothesis e_{CD} is invalid because this edge intersects true edge e_{AB}. Both contexts are eliminated, resulting in the context graph shown in Figure 5.8(b).

The effect is that when a grouping is confirmed those hypotheses that violate its elements (either feature or match) are eliminated. In the final set of matches in the root context there will be no contradictions. These matches correspond to a solution to the stereo matching problem.

As described earlier, matching is defined recursively, so that when two features at a level are matched, so are their constituent features at the lower levels. Matching at one level of the hierarchy affects matching at lower levels. For example, consider the effect of surface matches on line matches. When two surfaces are matched, the line segments constituting their boundaries are also

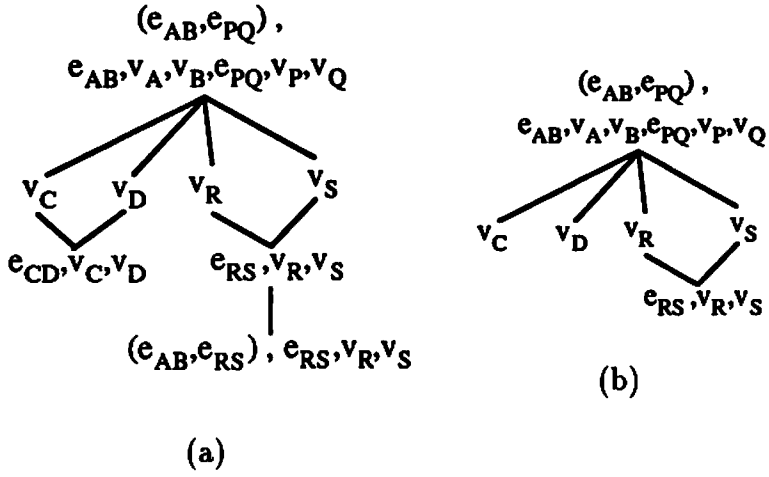


Figure 5.8: The effect of confirming the context with the edge-match hypothesis (e_{AB}, e_{PQ}) (of Figure 5.2) is shown in this figure. (a) Intermediate stage. (b) Final stage.

matched. When the surface match is confirmed so are the corresponding line matches. This affects the matching at the line level in three ways. First, there are fewer lines left to be matched at the line level. Second, any new line match that violates confirmed line matches is eliminated by the TMS. Third, the confirmed line matches serve as seeds for match groupings at the line level.

5.5 Grouping Constraints

Two classes of groupings result during the grouping and confirming procedure described in the previous section. *Direct* groupings are those that result when hypotheses are grouped directly based on structural and perceptual relations. *Indirect* groupings are those that result when contexts are confirmed. The underlying hypotheses of confirmed contexts are asserted into the root and are inherited down the context hierarchy into all the contexts. These hypotheses are then grouped with the other hypotheses in these contexts.

Certain groupings of features or matches are contradictory. Our objective is to eliminate all contradictory groupings (whether direct or indirect). As an example of a contradictory grouping, consider the intersecting edge hypotheses e_{AB} and e_{CD} shown in Figure 5.2. Physical edges in the scene cannot intersect, so these edge hypotheses are mutually contradictory. Of course, this is not to say that two edges in a scene cannot cross each other when viewed from some viewpoint. But if they do, part of one edge will be occluded by the surface formed by the other and this edge will not be detected by the feature grouping process. Any context with a grouping of intersecting edge hypotheses is therefore a *nogood.* Such *nogood* contexts are specified to the system explicitly using *nogood pattern combinations* (nogoods, for short). These are sets of patterns that identify contradictory groupings of hypotheses. This is an elegant way of specifying constraints. Any context with propositions that match all the patterns of a nogood is eliminated by the TMS. The effect is that in the final solution there are no contradictions. As an example, the situation corresponding to the intersecting edges in Figure 5.2 is identified with the following nogood:

```
(nogood
    (edge-hypothesis ?eX)
    (edge-hypothesis ?eY)
    (intersects ?eX ?eY))
```

The patterns match propositions in the database. The TMS uses this specification to identify those contexts that have matching propositions for all the patterns in this nogood and eliminates them. The nogood shown above can be classified as a *topological* constraint (see the following paragraph).

Below we illustrate some of the other grouping constraints used in the system.

uniqueness constraint: Any feature can match at most one feature. This follows from the fact that the image of a point in the scene will project onto a single point in each view. The uniqueness constraint for edge matches is illustrated in Figure 5.9, where the edge match hypotheses (e_A, e_P), and (e_A, e_R) cannot be grouped because e_A can match only one edge. This constraint is specified as the following nogood:

```
(nogood
   (edge-match ?eX ?eY)   ;; if ?eX matches ?eY
   (edge-match ?eX ?eZ))  ;; and also matches ?eZ
```

This constraint needs to be modified for lines, which tend to be fragmented.

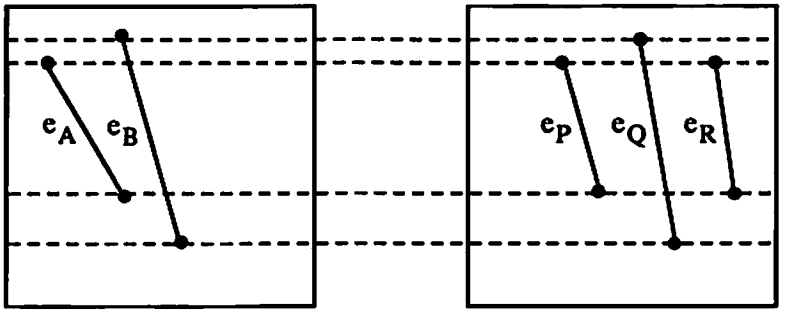


Figure 5.9: Uniqueness and ordering constraints for edges.

In this case, if a line matches more than one line, we expect those matches to be collinear. This is illustrated in Figure 5.10, where l_A can match both l_B and l_C.

ordering constraint: This constraint was originally used by Baker and Binford [71] and precludes any order reversal when matching along an epipolar

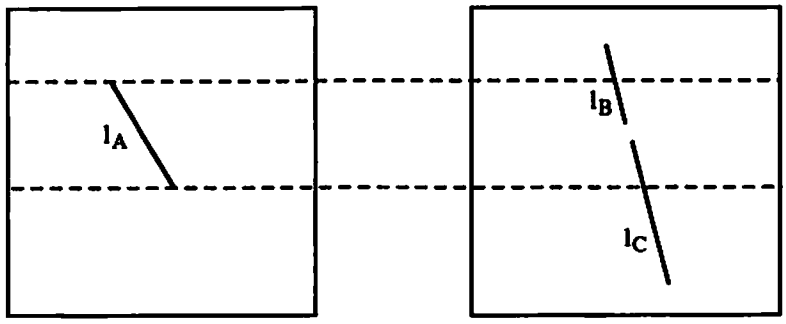


Figure 5.10: Uniqueness constraint for lines.

line. For example, in Figure 5.9, (e_A, e_R) and (e_B, e_Q) are inconsistent because e_A is to the left-of e_B, but e_R is to the right-of e_Q. The ordering constraint is satisfied in most natural scenes. Wires and overhanging surfaces may, however, create problems [71]. Most stereo systems enforce this constraint strictly [71, 74, 47]. The advantage is that the constraint reduces the search space drastically at the expense of missing a few correct matches. The nogood representation of the ordering constraint for edge matches looks like this:

```
(nogood
   (edge-match ?e_A ?e_P)
   (left-of ?e_A ?e_B)
   (edge-match ?e_B ?e_Q))
   (right-of ?e_P ?e_Q)   ;; order reversal
```

topological constraints: The topology of the feature hierarchy is viewpoint insensitive and we do not expect it to change between scenes. Topological constraints rule out several contexts. As one example, consider the matches (e_A, e_P) and (e_B, e_Q) in Figure 5.11. The matches are connected in one

scene and disconnected in the other. So their grouping is inconsistent. Similarly, the edge match hypothesis (e_C, e_R) is incompatible with the edge hypotheses e_D and e_S because these edge hypotheses do not match and we expect topology to be preserved across scenes. The vertex hypothesis v_{AB} and the match hypotheses (l_A, l_P) and (l_B, l_Q) are incompatible because l_P and l_Q do not intersect on the epipolar line of v_{AB} when extended. The last two examples illustrate ternary constraints involving three hypotheses. Such higher order constraints are easily incorporated when using a ATMS. Topological constraints are also used during the feature grouping process. For example, intersecting edge hypotheses form a nogood combination. Similarly, a vertex hypothesis is incompatible with any edge hypothesis that it lies on (if it is not a vertex of that edge).

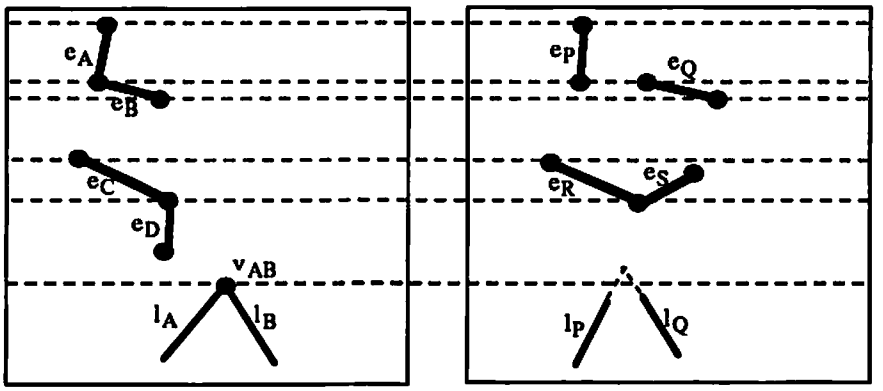


Figure 5.11: Topological constraints.

The constraints described in This section are binary (in some cases, ternary) constraints [52] that prohibit certain groupings of matches. The epipolar, orientation, length, area, and contrast constraints described in Section 5.3 are unary

constraints that help in reducing the number of matches that are hypothesized and later subjected to the grouping process.

5.5.1 Matching Complexity

In this section we do an approximate analysis of the matching complexity. We start by analyzing the complexity at one level of the hierarchy and later analyze the multi-level case. Suppose there are N features at a particular level of the hierarchy and E_l epipolar lines in each image. The number of features per epipolar line is

$$\frac{N}{E_l}.$$

Features are matched across epipolar lines. So the number of feature pairs to be tested for matching across each epipolar line is

$$\frac{N^2}{E_l^2}.$$

Since there are E_l of these epipolar lines, the number of feature pairs to be tested for the entire image is

$$\frac{N^2}{E_l}. \tag{5.1}$$

Suppose a fraction f of these satisfy the orientation and contrast constraints. Then the final number of matches will be

$$M = f\frac{N^2}{E_l}.$$

We now group these match hypotheses based on structural and perceptual relations. For simplicity of analysis, assume that there are $\frac{M}{3}$ groupings of size 3. Binary and ternary constraints are evaluated for each grouping. The complexity of these constraint evaluations is proportional to

$$\frac{M}{3} = \frac{fN^2}{3}. \tag{5.2}$$

These groupings are confirmed at random. When a grouping is confirmed all matches comprising the grouping are converted to truths and asserted into the root context. These matches are then inherited down the context tree and they must be compared with the other matches outside the root context to check for violations of grouping constraints. The first confirmation leaves $(M-3)$ match hypotheses outside the root context, the second $(M-6)$ and so on, till all groupings are confirmed (assume for simplicity that no grouping is eliminated because of constraint evaluations). The complexity of the confirmation process is proportional to

$$(M-3)+(M-6)+(M-9)+\ldots = \frac{M^2}{6} - \frac{M}{2} \approx \frac{M^2}{6} = \frac{f^2N^4}{6{E_l}^2}. \tag{5.3}$$

From (5.1), (5.2) and (5.3) it can be seen that the time for matching is essentially decided by (5.3), the confirmation process. This was also empirically observed.

The above analysis shows that the complexity of matching at each level is $O(N^4)$, where N is the number of features at that level. So if only lines were matched (i.e. there was no hierarchy) the complexity is $O(l^4)$, where l is the number of lines in each frame. We now examine how the hierarchical matching

procedure reduces the proportionality constant of this complexity (complexity will still be proportional to l^4). Let the reduction in number of feature hypotheses between consecutive levels be k_1. Let the fraction of reduction in the number of features (not hypotheses) between consecutive levels be k_2. Then

$$k_1 \leq k_2$$

because not all feature hypotheses may participate in the feature grouping process. Table 5.1 assists in computing the complexity reduction. The first column shows the names of features. The second column shows the number of features at each level, assuming there are l lines at the line level and a fractional reduction of k_1. The third column shows the number of features that are actually considered for matching. For example, consider the s_m surfaces matched at the surface level. These correspond to $\frac{l}{k_2}s_m$ edge matches, $\frac{l}{k_2^2}s_m$ vertex matches and $\frac{l}{k_2^3}s_m$ line matches. These matched features will not be considered for matching again. This leads to a reduction in the number of features participating in matching at lower levels. The third column shows the original number of features decremented by the features already matched at previous levels. So the hierarchical matching strategy reduces complexity by a factor of

$$\frac{{l_m}^4 + {v_m}^4 + {e_m}^4 + {s_m}^4}{l^4}$$

where the numerator sums the computations performed at different levels of the hierarchy. As a typical example, if $f = 0.6$ (60% of features at each level matched), $k_1 = 0.4$ (each level has 0.4 times the number of hypotheses at its lower

complexity analysis		
feature	*no. of features*	*no. to be matched*
lines	l	$l_m = l - \frac{f}{k_2{}^3} s_m - \frac{f}{k_2{}^2} e_m - \frac{f}{k_2} v_m$
vertices	$k_1 l$	$v_m = k_1 l - \frac{f}{k_2{}^2} s_m - \frac{f}{k_2} e_m$
edges	$k_1{}^2 l$	$e_m = k_1{}^2 l - \frac{f}{k_2} s_m$
surfaces	$k_1{}^3 l$	$s_m = k_1{}^3 l$

Table 5.1: Table for computing matching complexity

level) and $k_2 = 0.5$ (fractional reduction in number of features across two levels in the hierarchy) the complexity reduction factor is 0.01. So the hierarchical matching procedure significantly reduces matching complexity. Further, it also ensures that the resultant matches are more correct. Both factors are important.

We have not accounted for the additional complexity introduced by the feature grouping process. Suppose there are N features at one level. At most N^2 comparisons need to be made to determine the feature relations at the current level and to group features to form the next level of the hierarchy. Because of the bucketing techniques described in the Appendix, the number of comparisons is much less and so the complexity is significantly less than that for matching and can be neglected. Further, these features may be needed anyway for further processing - for example, in object recognition.

5.6 Stereo Experiments

The feature grouping and hierarchical matching process is first demonstrated on a blocks scene shown in Figure 5.12. This stereo pair was also used in [72, 45] and so results can be directly compared. Lines extracted from the image are shown in Figure 5.13. There are 273 lines in the right image and 264 in the left. One can see that the straight line extractor breaks up curved contours into straight line segments. This could create a problem during matching of curved contours, because break points can be different in two views, especially if the views are not close to each other. Note that curved contours need to be treated differently anyway. For example, the boundaries of curved surfaces like spheres appearing in an image are only apparent boundaries. These boundaries in the two views are images of different portions of the curved surfaces. So they need to be treated differently from planar surface boundaries [117]. We do not address the issue of curved surface reconstruction. We treat all boundaries as originating from real (rather than apparent) boundaries.

Vertices are shown in Figure 5.14. The dark circles are the vertices. Surface edges formed by grouping vertices are shown in Figure 5.15. Contiguous edges are grouped together to form surfaces. The entire hierarchical grouping process took approximately 23 minutes for both images. The algorithm was written in ART [5] and LISP and ran on a Texas Instruments LISP machine with 16MB memory. First, surfaces (edge-rings) in the images are matched. The matches are shown in Figure 5.16. Note that the system correctly matches the repetitive squares on the Rubik cube. This is made possible by the grouping of adjacent and parallel surface matches. Any grouping which violates the ordering constraint

is a *nogood* and will be eliminated by the TMS. So only one match grouping (the correct one) survives. Currently, the grouping in our system is only three deep (for example: at most three surface matches are combined together before confirmation). This could create problems with large repetitive structures (for example: several identical squares adjacent to each other and spanning the same set of scan lines). In such a case, the size of the match groupings must be increased.

Next, edges are matched using unmatched surfaces as foci of attention to form structural groupings. This stage looks for partial matches of surfaces. Results are shown in Figure 5.17. Final matches after matching at the edge level are shown in Figure 5.18. Matches after matching at the vertex level are shown in Figure 5.19. Unmatched vertices are used as foci of attention to derive the matches shown in Figure 5.20. The final matches after matching at the line level are shown in Figure 5.21. The entire hierarchical matching process took 50 minutes. Figures 5.22 and 5.23 show labelled vertex and line matches. Matching vertices are given the same label in Figure 5.22 and matching lines are given the same label in Figure 5.23. Features in the left image are labelled with their symbolic name and those in the right are labelled with the symbolic names of their matches. This is similar to the way matching results are presented in [46]. Collinear lines in the right image may match the same line in the left image and hence may have the same label. Also, a line in the right image may match collinear lines in the left image. In that case, we randomly choose one of these collinear lines for labelling the line in the right image.

Next, the stereo algorithm is illustrated for the building scene shown in Figure 5.24. Extracted lines are shown in Figure 5.25 and vertices in Figure 5.26.

These vertices are grouped into edges (Figure 5.27). No surfaces derived from the images match exactly. However the unmatched surfaces serve as foci of attention when matching. Results after matching edges are shown in Figure 5.28. The vertices and lines of matched edges are also matched. These matches constrain the matches at lower levels. For example, any line match hypothesis that violates the ordering constraint with respect to these confirmed line matches will be negated. Figure 5.29 shows the feature matches after matching at the vertex level. Results after matching lines by focusing on unmatched vertices are shown in Figure 5.30. The final result after matching lines is shown in Figure 5.31. Figures 5.32 and 5.33 show labelled vertex and line matches.

Next, we illustrate the matching process for a stereo pair of images of a portion of LAX airport (Figure 5.34). Figures 5.35, 5.36 and 5.37 show the feature hierarchy. Results of matching surfaces are shown in Figure 5.38. Figure 5.39 shows the results after matching edges using unmatched surfaces as foci of attention. Final edge matches are shown in Figure 5.40. Vertex matches are shown in Figure 5.41. Figure 5.42 shows the final matches at the line level. Figures 5.43 and 5.44 show labelled vertex and line matches.

5.7 Motion correspondence

The ideas used in stereo matching can also be used in motion correspondence. In that case we get both point and line matches. Point matches result from vertex matches and line matches result from line segment matches at the lowest part of the hierarchy. Note that surface and edge matches finally reduce to vertex and line matches. Our objective is to track these matches over several frames.

This is accomplished by matching two adjacent frames at a time and maintaining symbolic links across the entire image sequence. These matches can then be used in any point or line based motion estimation systems [77, 78, 79, 80, 81, 82, 83, 85, 86, 48, 49, 92]

Smoothness of disparity is used as a criterion when deriving point matches in [88, 89, 90] and line matches in [44]. However this is a weak constraint, because interest points and lines in the scene are usually not contiguous but are in fact disconnected and sparsely distributed. Such points need not have similar disparity. We attempt to derive stronger constraints based on the topology of the scene. The algorithm used for motion correspondence is similar to the stereo matching algorithm. However, the epipolar constraint cannot be used during matching. We also did not use the ordering constraint. Since the images are from a closely sampled sequence we expect adjacent images to have a disparity less than a threshold. Disparity here refers to the relative distance between the centroids of features to be tested for matching, using one image as the reference. This threshold depends on how closely the images are sampled. The closer the images the smaller the disparity threshold. We also track the features across the entire image sequence by retaining the matches over the sequence. This results in point and line matches. Point matches result from vertex correspondences and line matches through line segment correspondences. Matching complexity is still $O(l^4)$ and much of the complexity analysis for stereo matching in the previous section applies here. The only difference is that the number of match hypotheses at one level of the hierarchy is on the average $M = fN^2$, where N is the number of features at that level and f is the fraction of the total possible matches (N^2) that satisfy the disparity constraint.

5.8 Motion Experiments

The feature grouping and hierarchical matching process is first illustrated graphically for the first two frames of a nine frame office sequence. A Texas Instruments Explorer lisp machine with 16MB memory was used for running the algorithm. The first two frames are shown in Figure 5.45. Edges in the image are first detected using the Canny edge detector [102]. This process took about 2 minutes of user time for each image. Lines in the image are then extracted by linking contiguous edge pixels using the algorithm described in [6]. This took about 2 minutes and 30 seconds for each image. The lines extracted are shown in Figure 5.46. These lines serve as input to the motion correspondence system. A feature hierarchy is constructed in each frame using this input. This took about 3 minutes 30 seconds for each frame. Vertices are shown in Figure 5.47. The dark circles are the vertices. Surface edges formed by grouping vertices are shown in Figure 5.48 as dark lines. Connected edges form surfaces. First, surfaces (edge-rings) in the images are matched. The matches are shown in Figure 5.49. Each surface in this figure is labelled at its centroid. Matching surfaces have the same label. Matches at edge level are shown in Figure 5.50 and those at vertex level are shown in Figure 5.51. The final line matches after matching at line level are shown in Figure 5.52. The entire matching process took 6 minutes of user time.

Figure 5.53 shows the fifth and ninth frames in the sequence. Figure 5.54 shows vertices in the fifth and ninth frames that were tracked from the first frame. These vertices are labelled by their symbolic name in the first frame. Similar, Figure 5.55 shows the lines that were tracked from the first frame. Figure 5.56 shows the trajectories of vertices that originate in the first frame and that were

tracked over 9 frames. These trajectories are shown superposed over the first and the ninth frames.

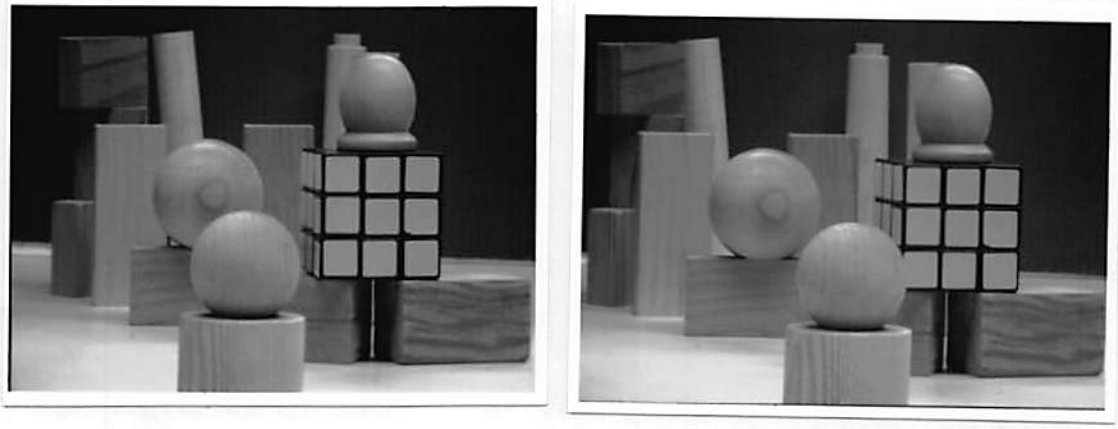


Figure 5.12: Right view and left view of a blocks scene.

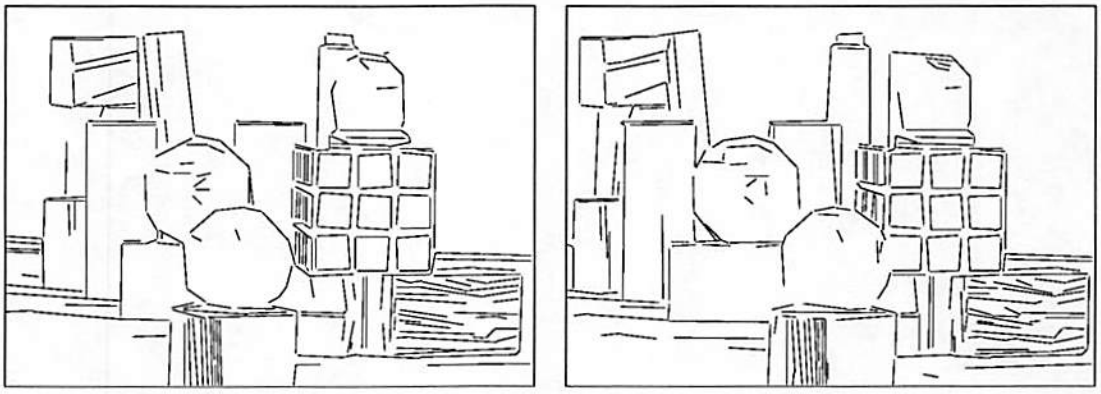


Figure 5.13: Lines extracted from the two views.

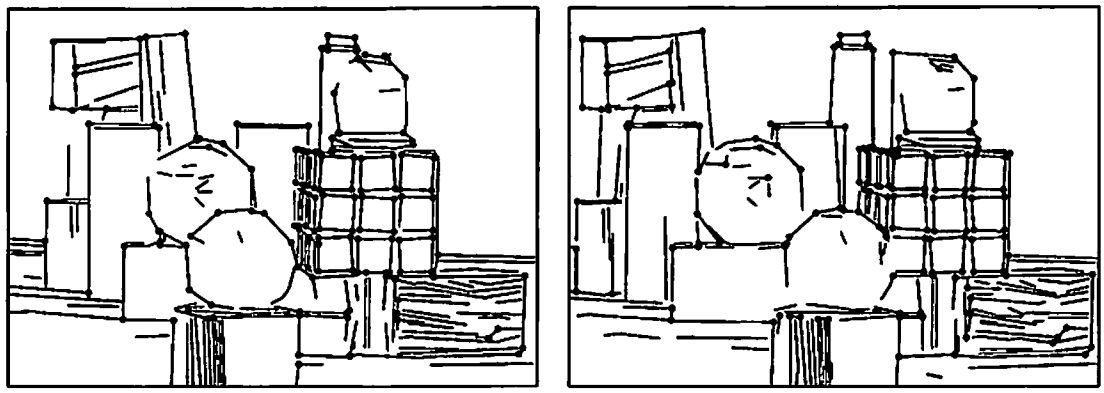


Figure 5.14: Vertices.

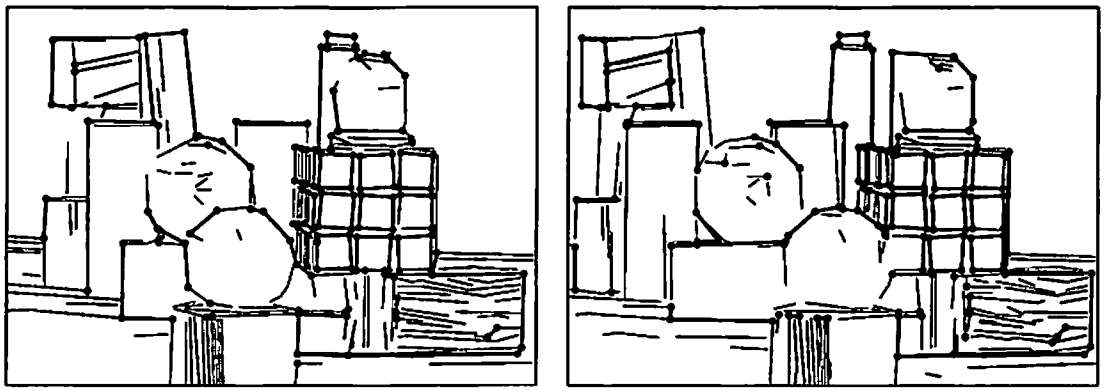


Figure 5.15: Edges (dark lines).

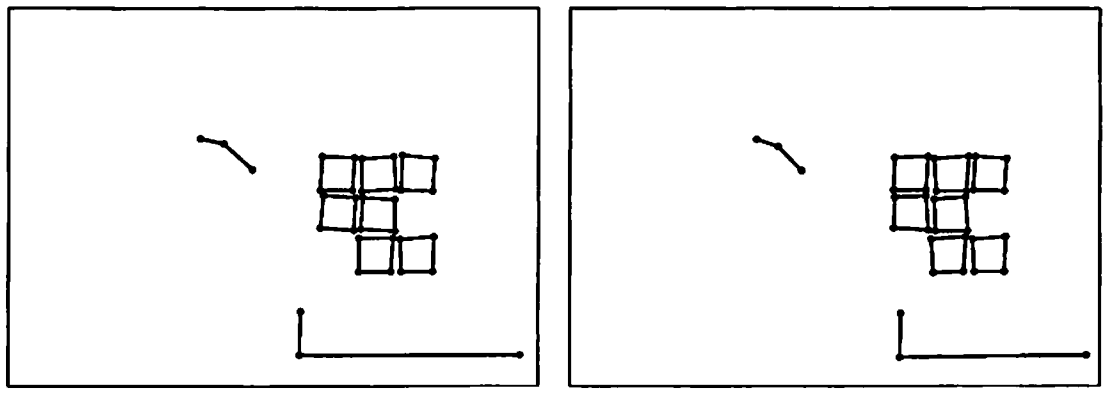


Figure 5.16: After matching at the surface level.

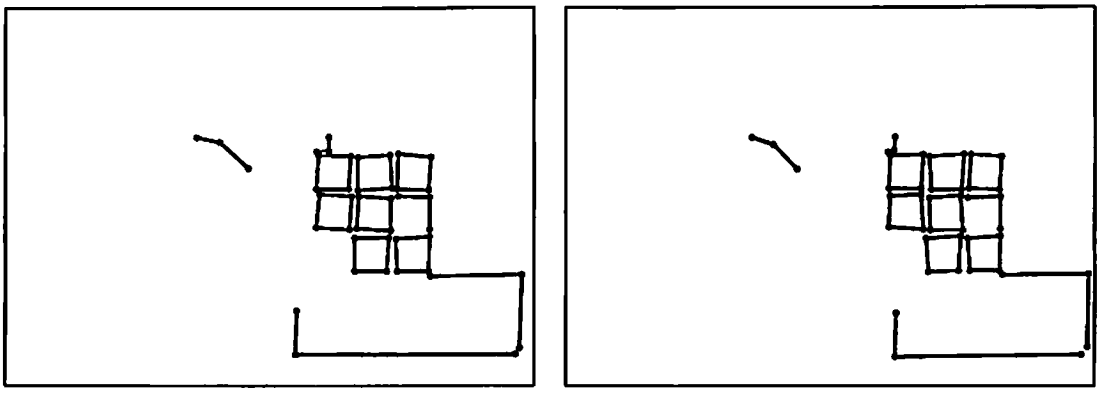


Figure 5.17: After matching edges using unmatched surfaces as foci of attention.

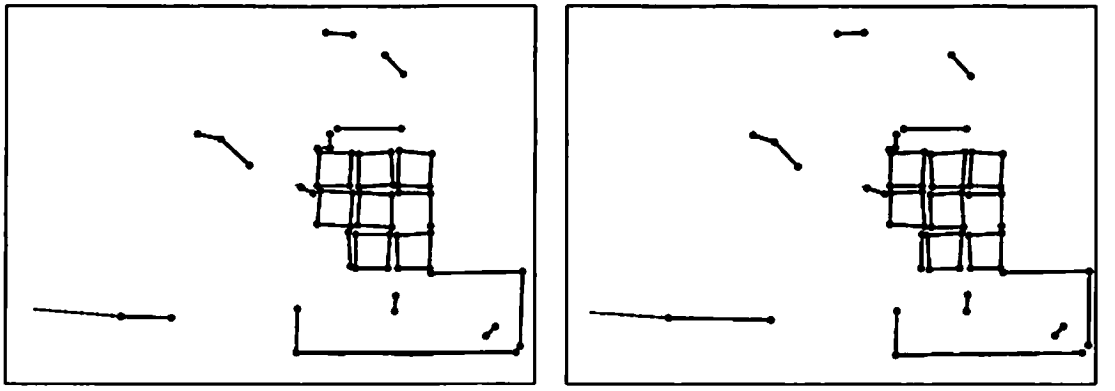


Figure 5.18: Final matches at the edge level.

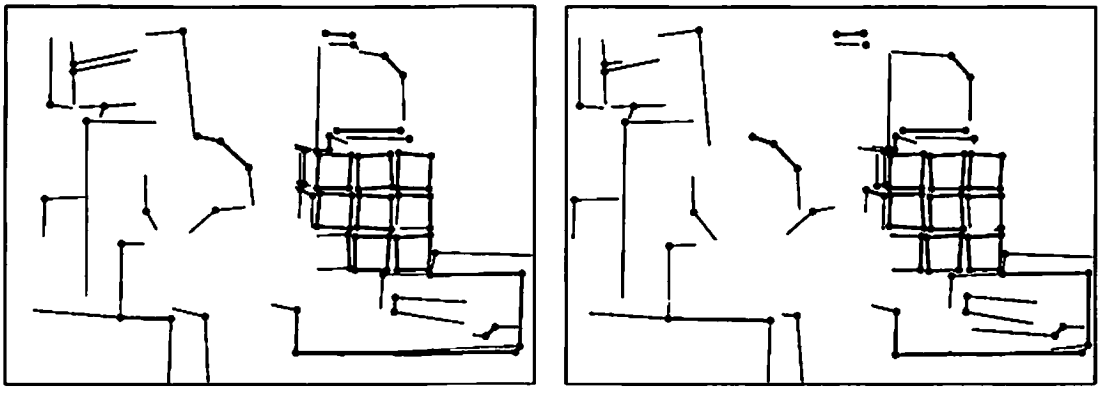


Figure 5.19: After matching at the vertex level.

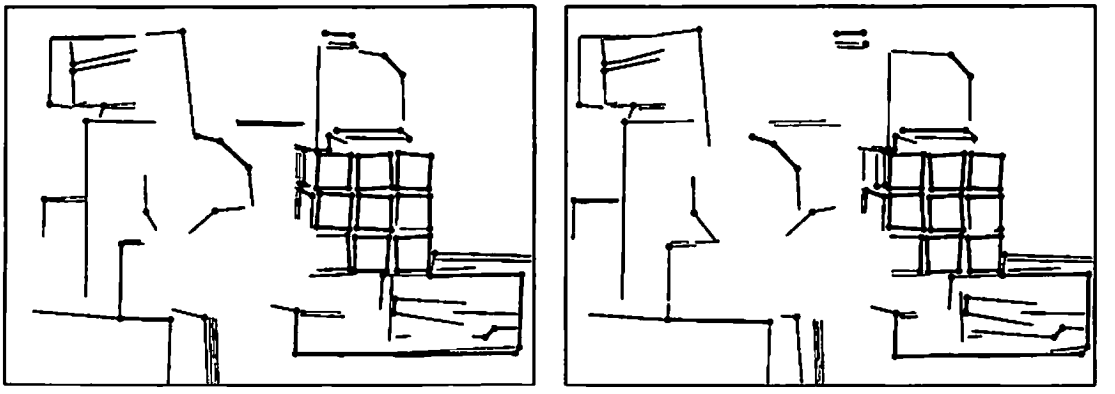


Figure 5.20: After matching lines using vertices as foci of attention.

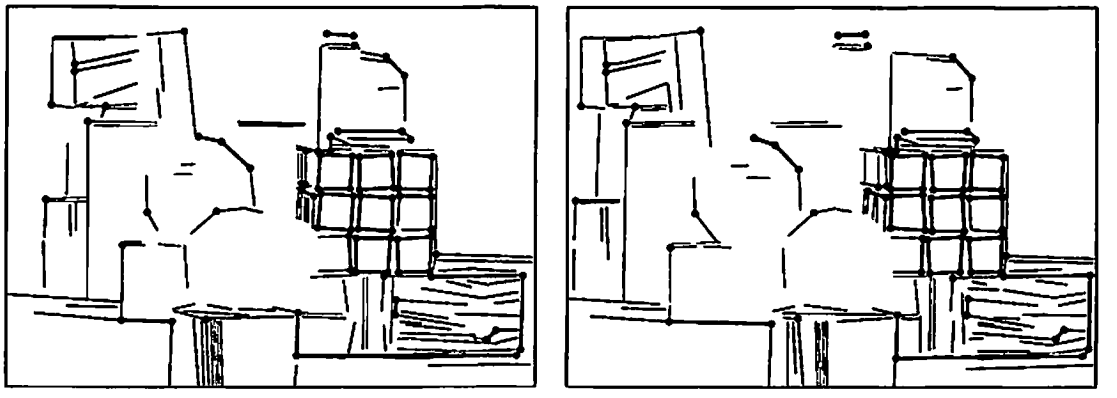
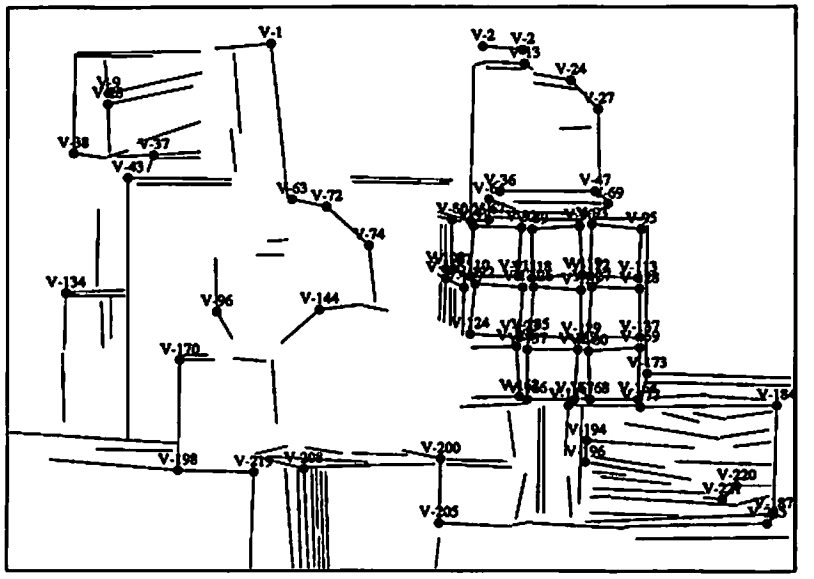
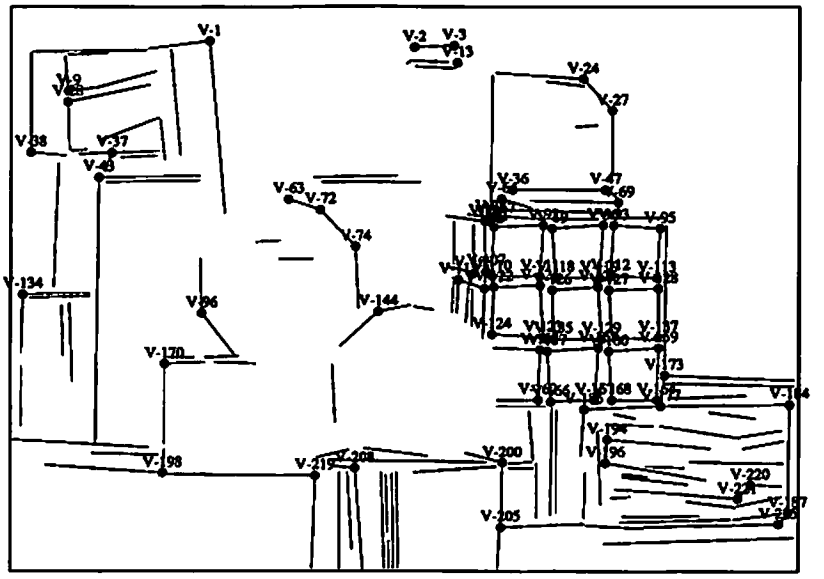


Figure 5.21: Final matches at the line level.

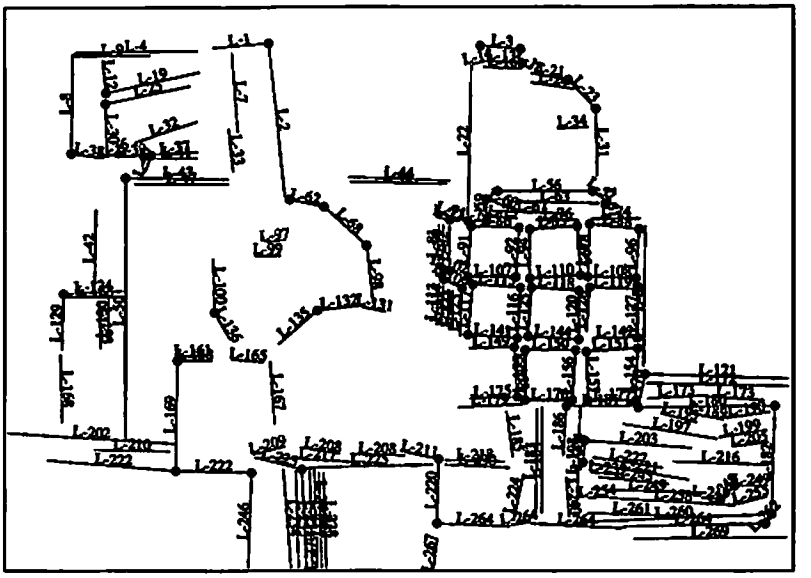


(a)

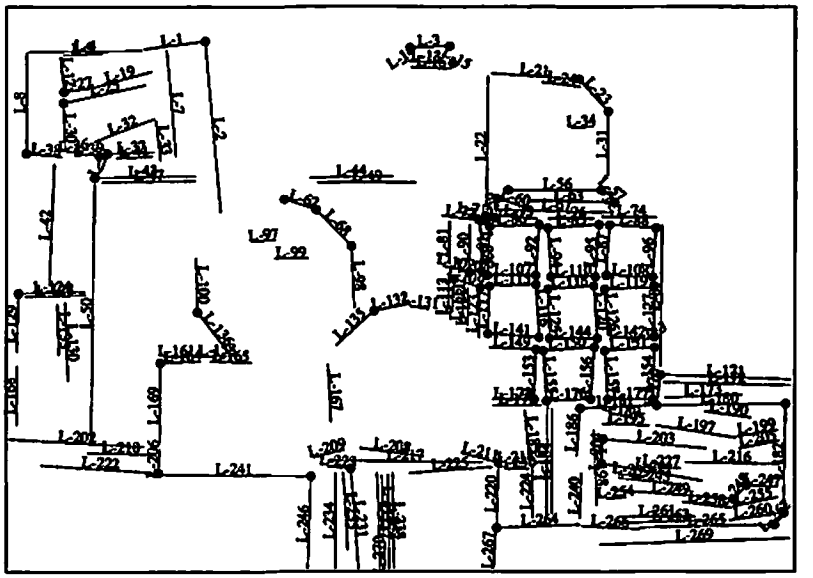


(b)

Figure 5.22: Vertex and line matches with vertex labels. (a) Right image. (b) Left image.



(a)



(b)

Figure 5.23: Vertex and line matches with line labels. (a) Right image. (b) Left image.

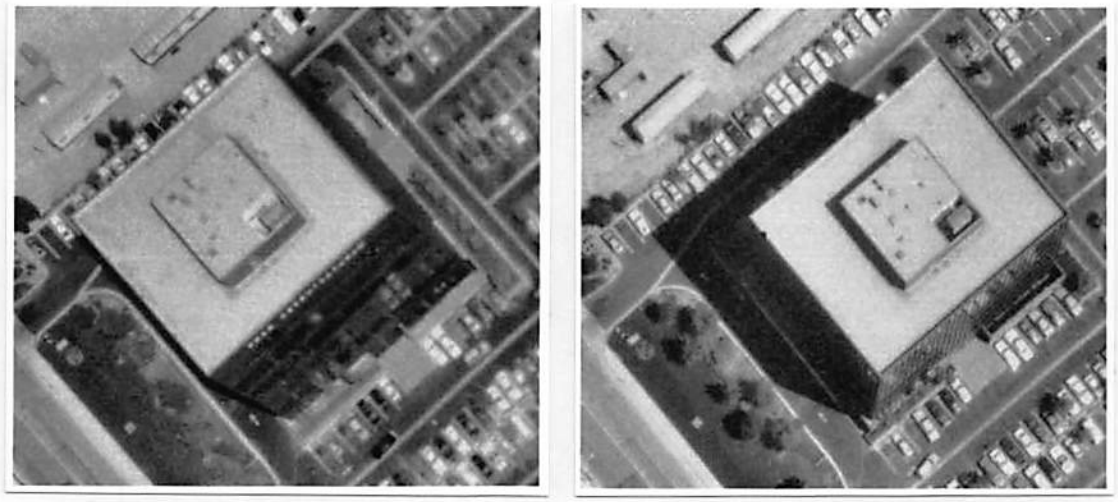


Figure 5.24: Right view and left view of a building image.

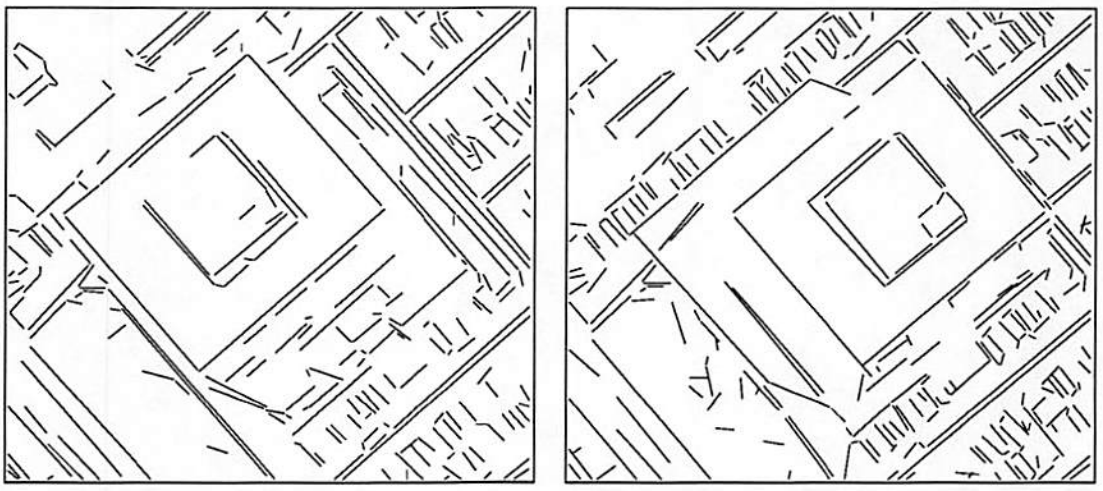


Figure 5.25: Lines extracted from the two views.

Figure 5.26: Vertices.

Figure 5.27: Edges (dark lines).

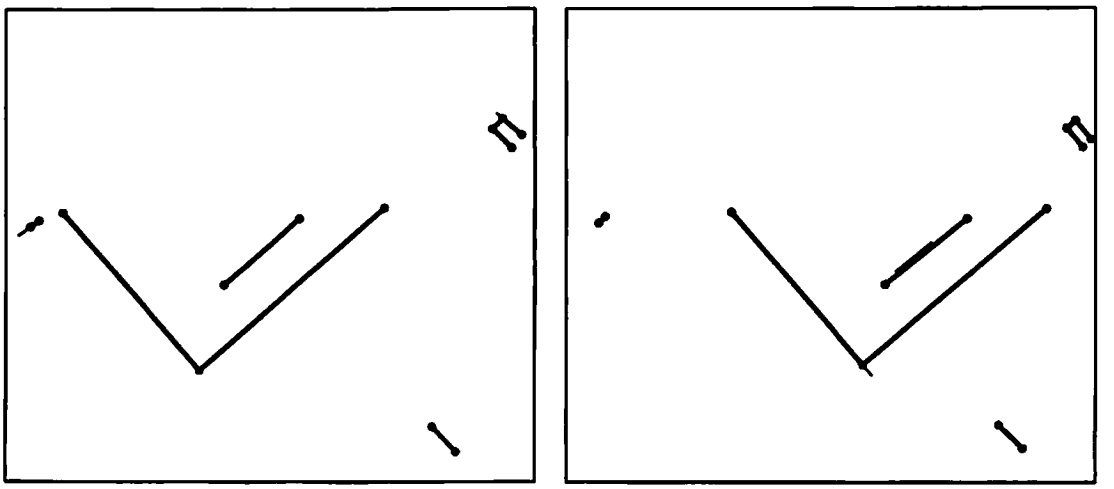


Figure 5.28: After matching at the edge level.

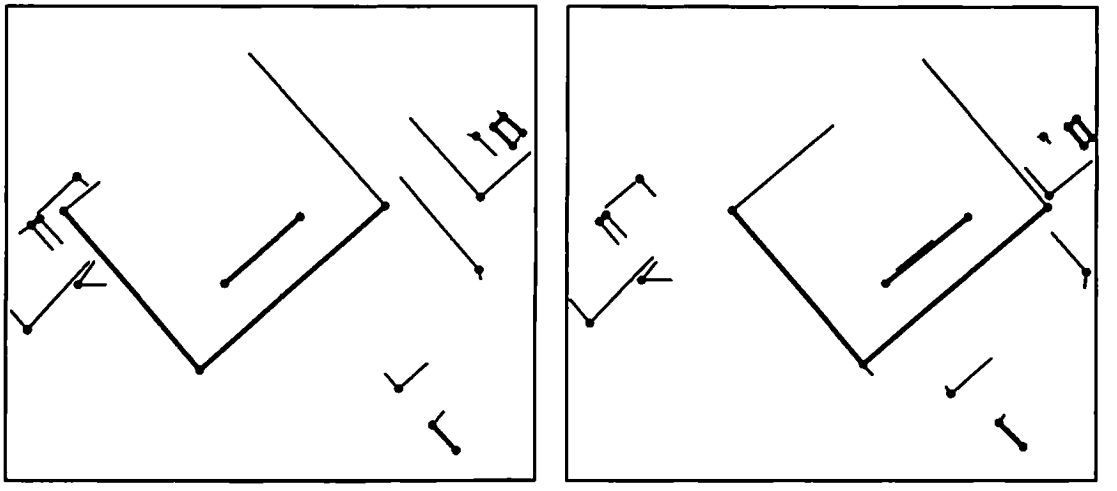


Figure 5.29: After matching at the vertex level.

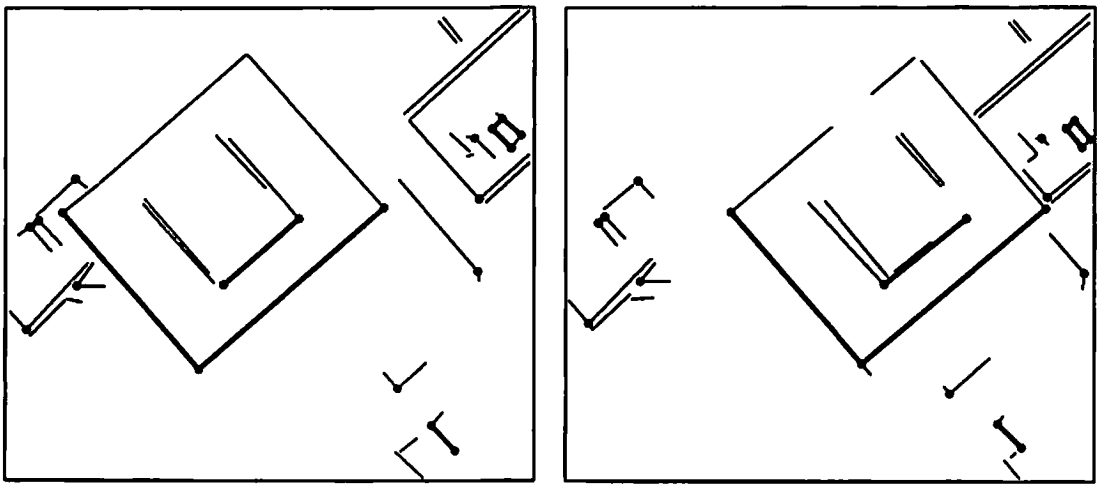


Figure 5.30: After matching lines by using unmatched vertex features as foci of attention.

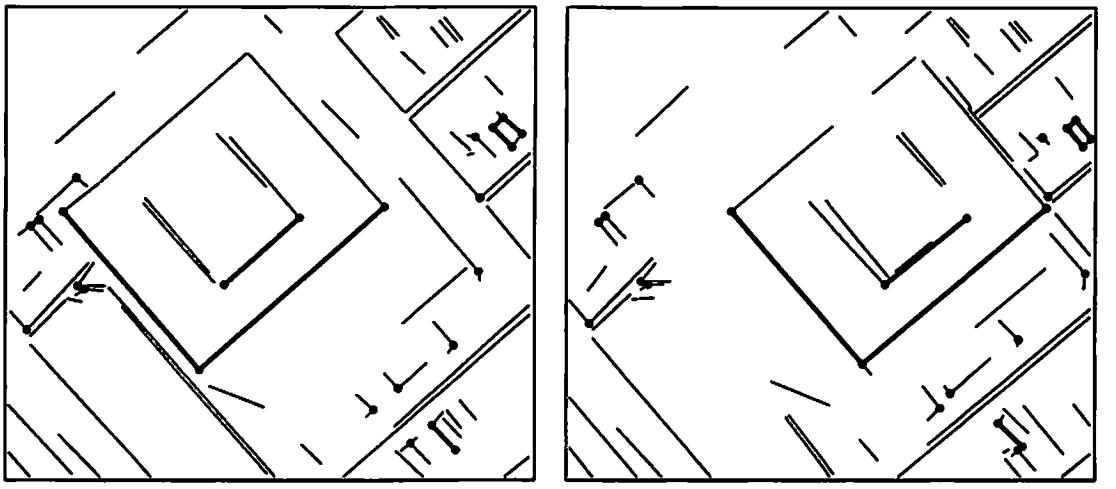
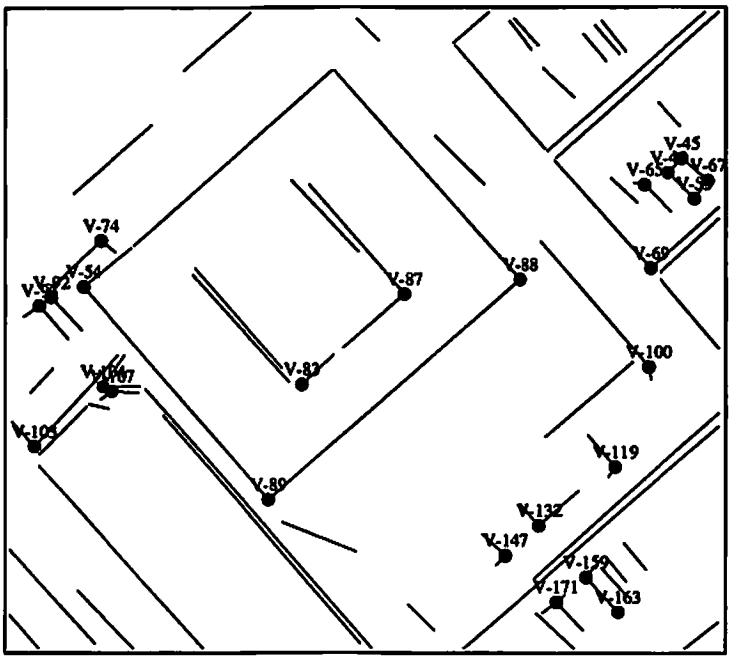
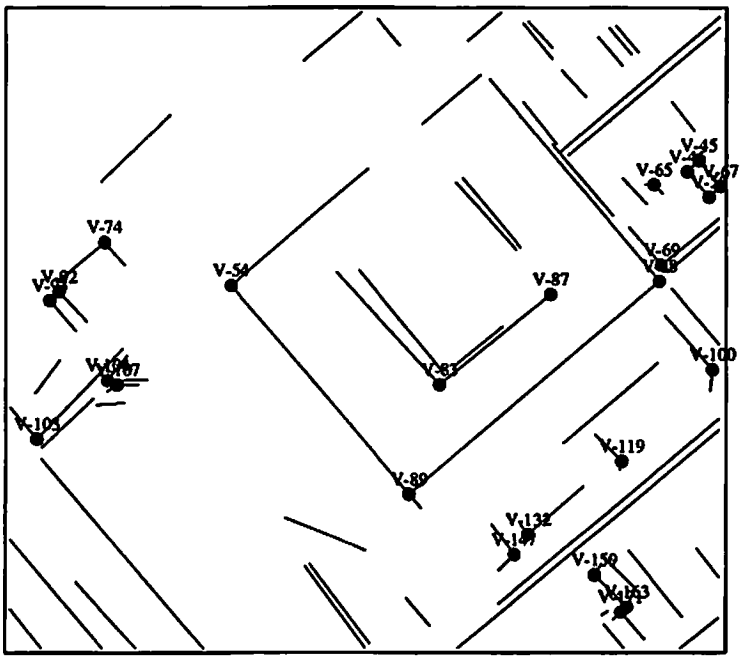


Figure 5.31: Final matches.

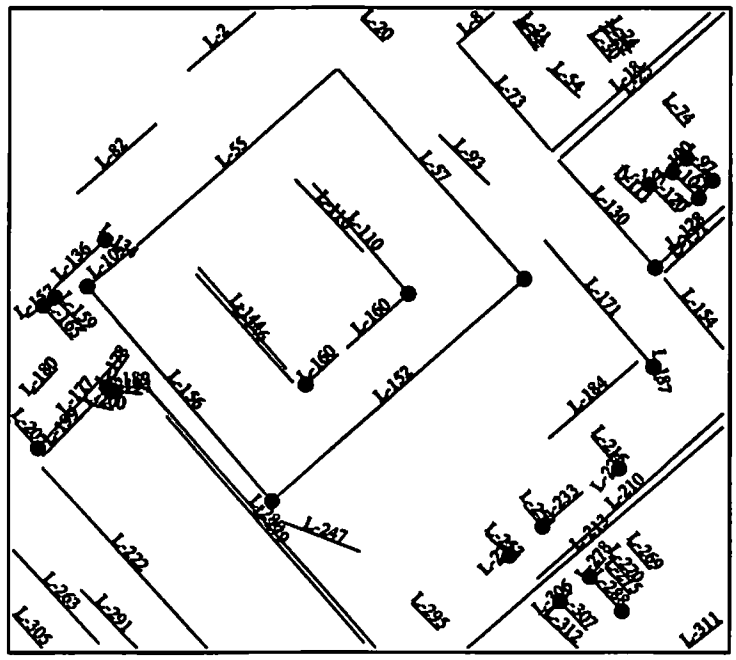


(a)

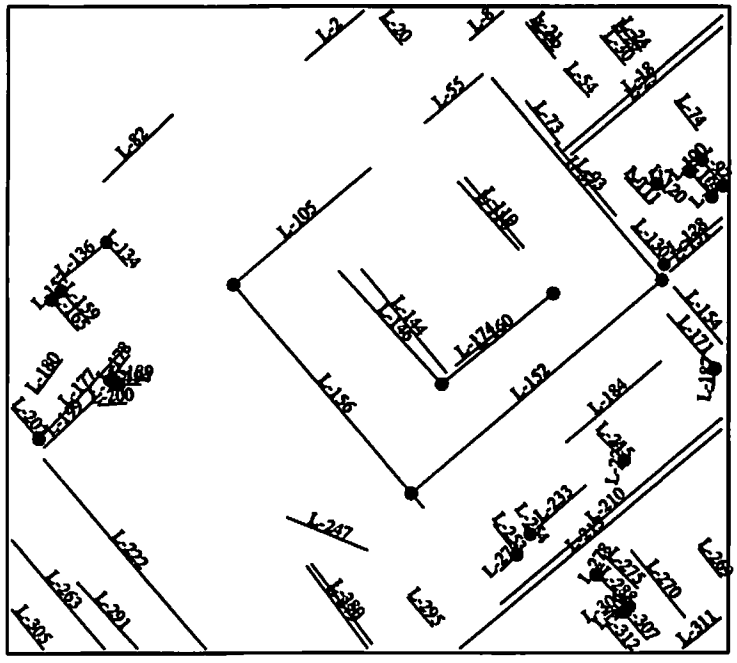


(b)

Figure 5.32: Vertex and line matches with vertex labels. (a) Right image. (b) Left image.



(a)



(b)

Figure 5.33: Vertex and line matches with line labels. (a) Right image. (b) Left image.

Figure 5.34: Right view and left view of a portion of LAX airport.

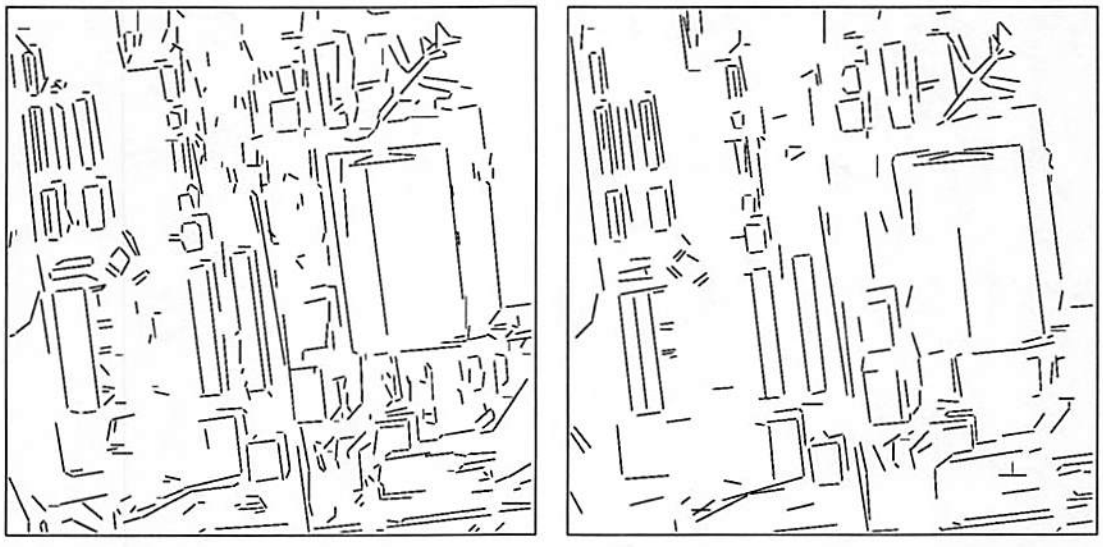


Figure 5.35: Lines extracted from the two views.

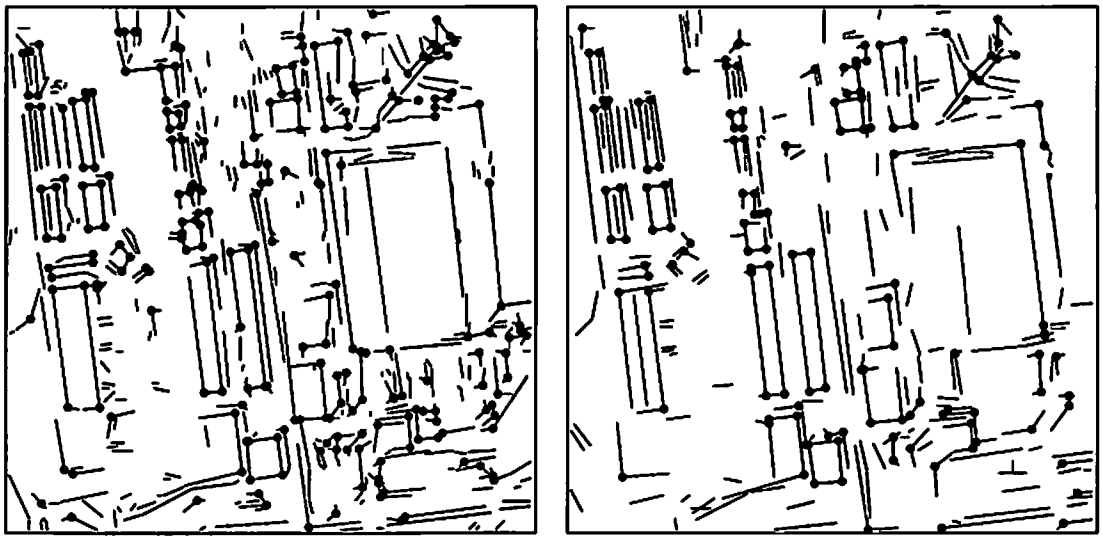


Figure 5.36: Vertices.

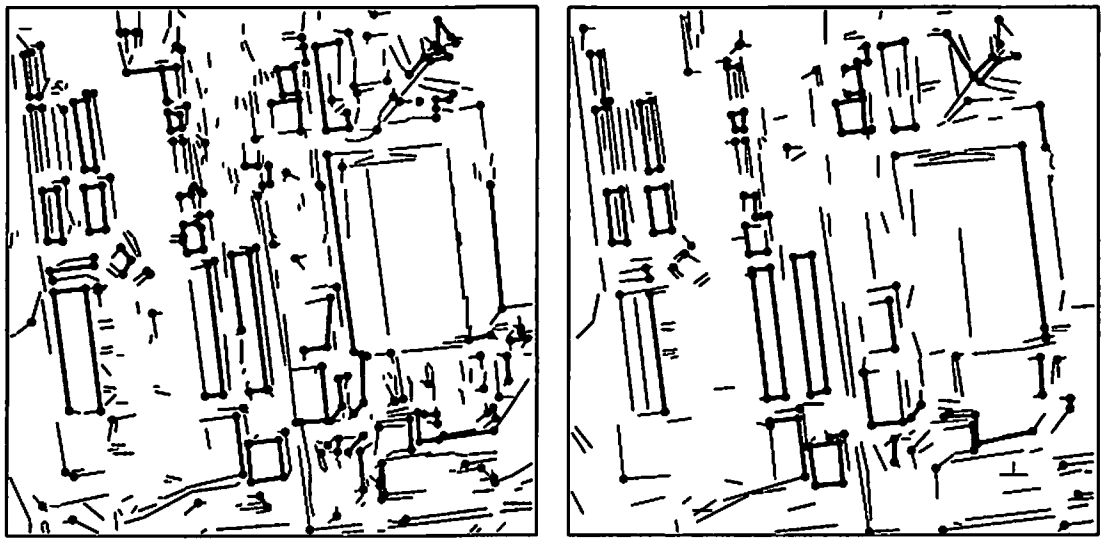


Figure 5.37: Edges (dark lines).

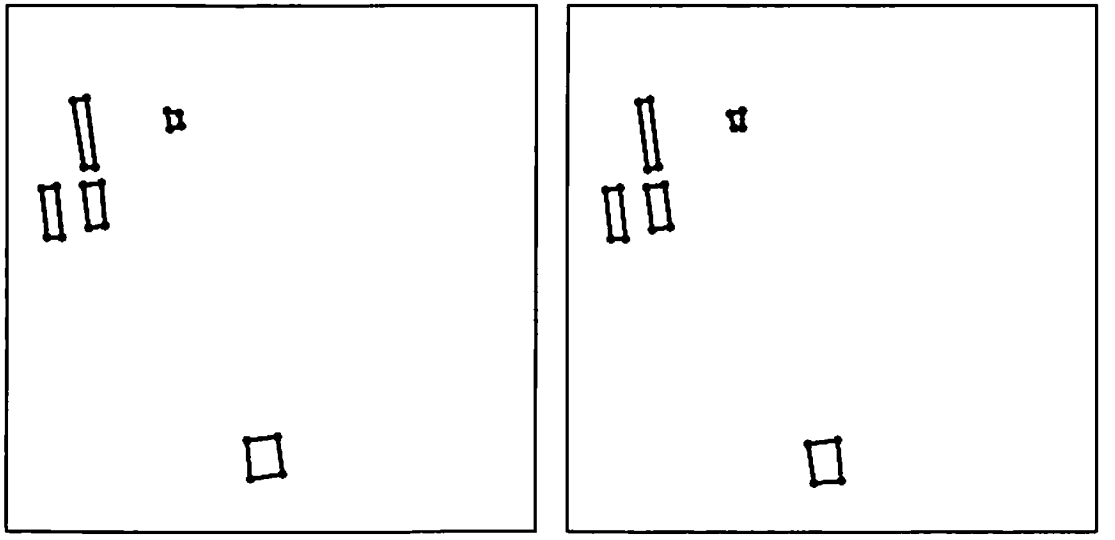


Figure 5.38: After matching at the surface level.

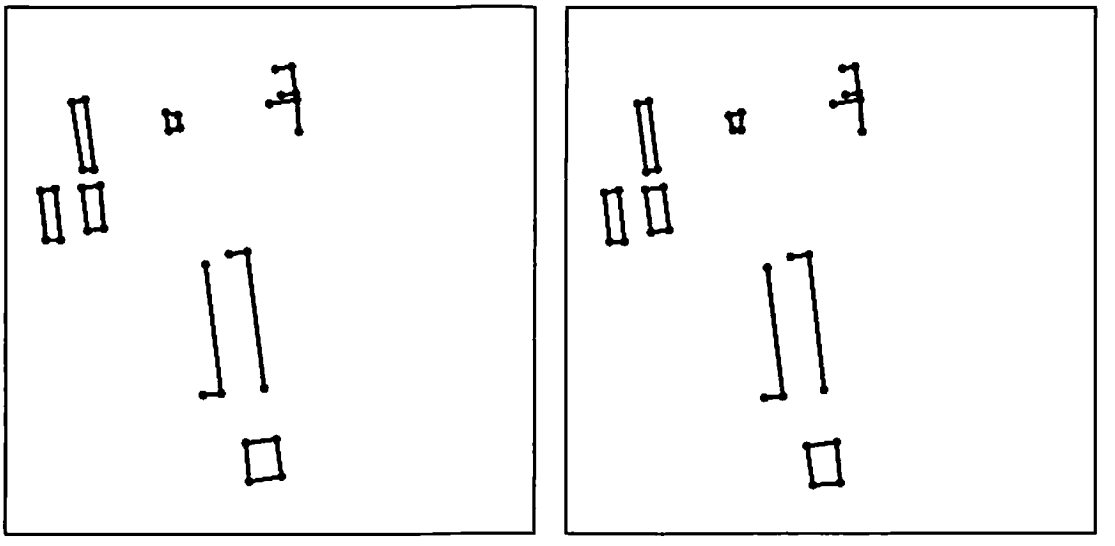


Figure 5.39: After matching edges using unmatched surfaces as foci of attention.

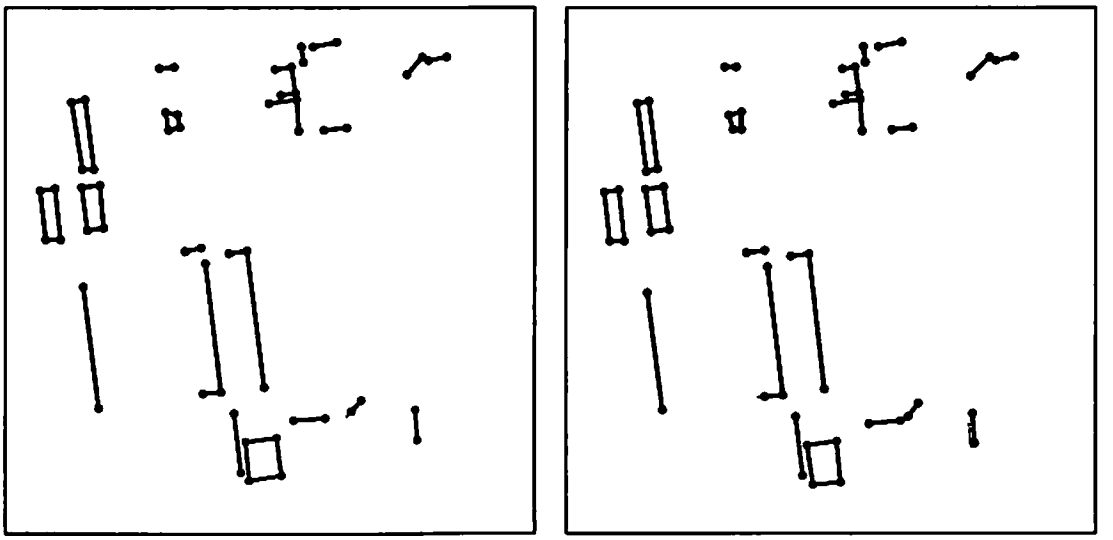


Figure 5.40: Final matches at the edge level.

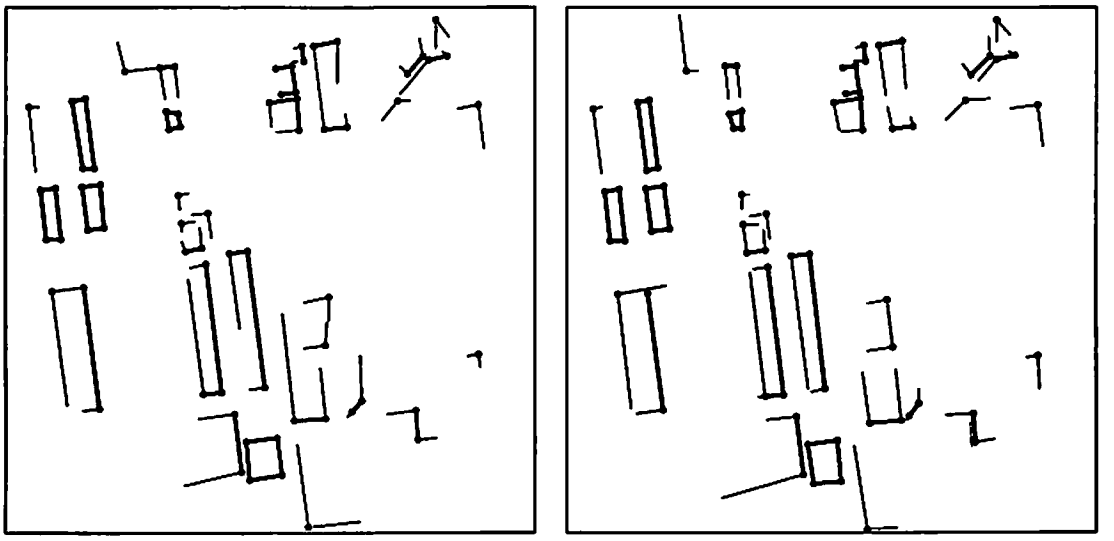


Figure 5.41: After matching at the vertex level.

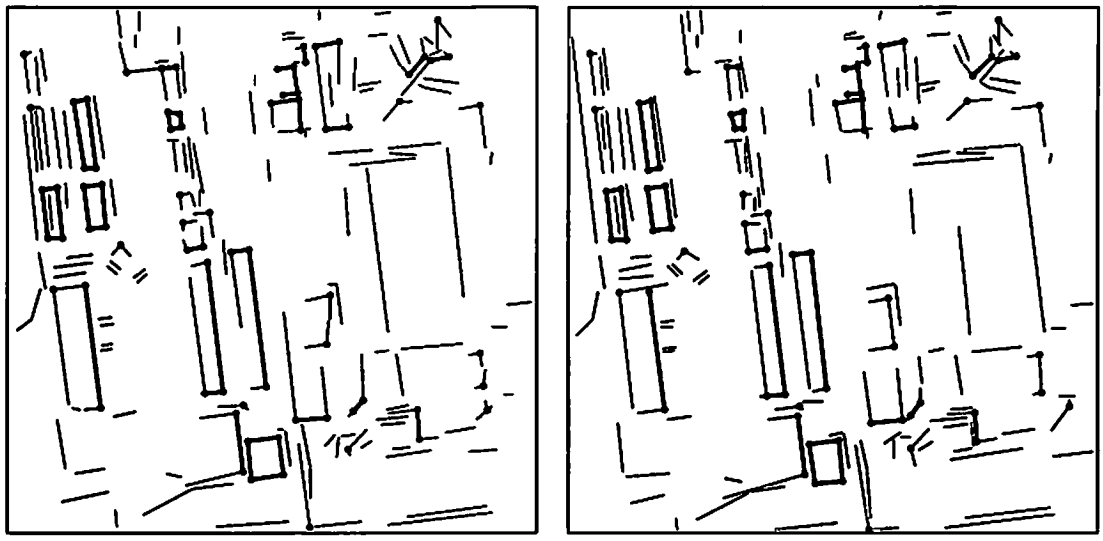
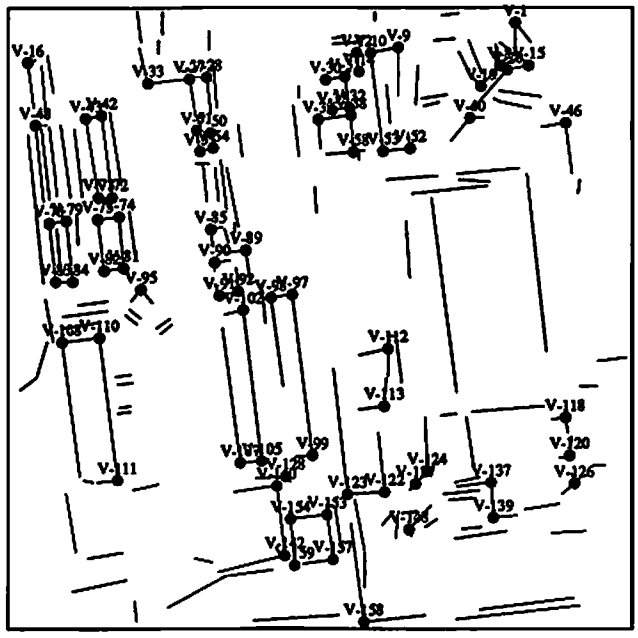
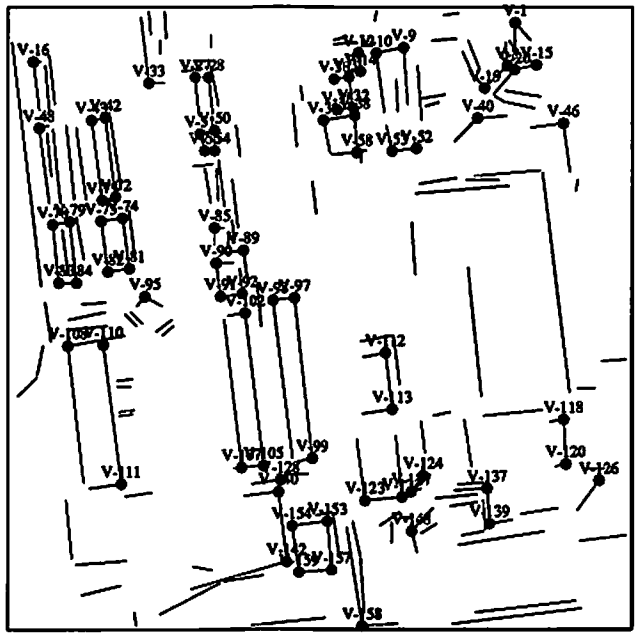


Figure 5.42: Final matches at the line level.

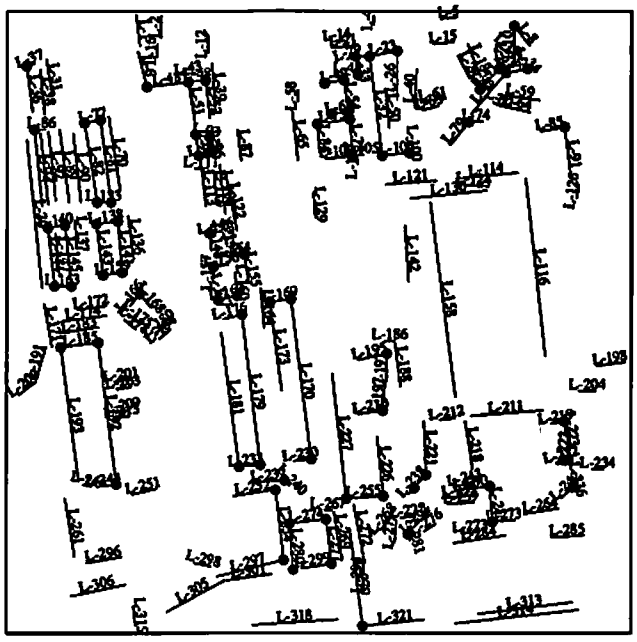


(a)

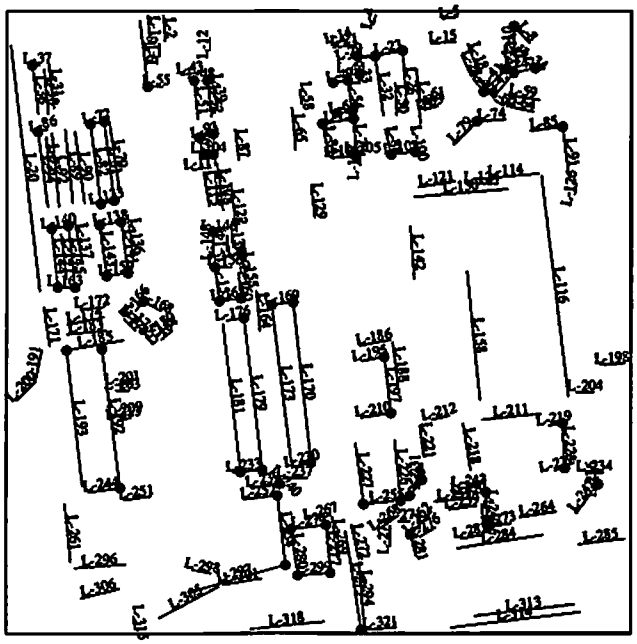


(b)

Figure 5.43: Vertex and line matches with vertex labels. (a) Right image. (b) Left image.



(a)



(b)

Figure 5.44: Vertex and line matches with line labels. (a) Right image. (b) Left image.

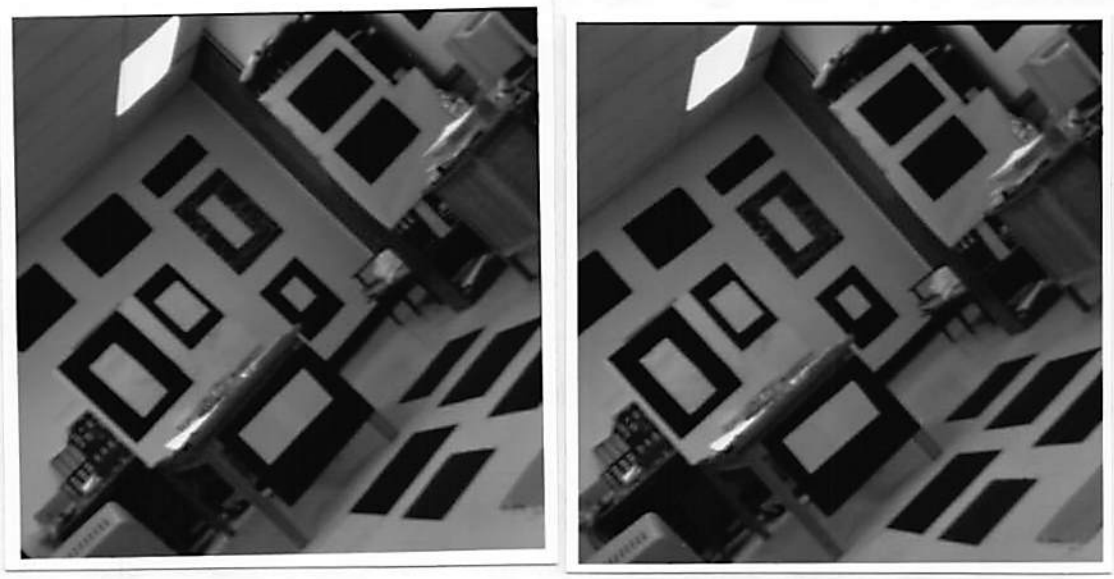


Figure 5.45: First and second frames in the office sequence.

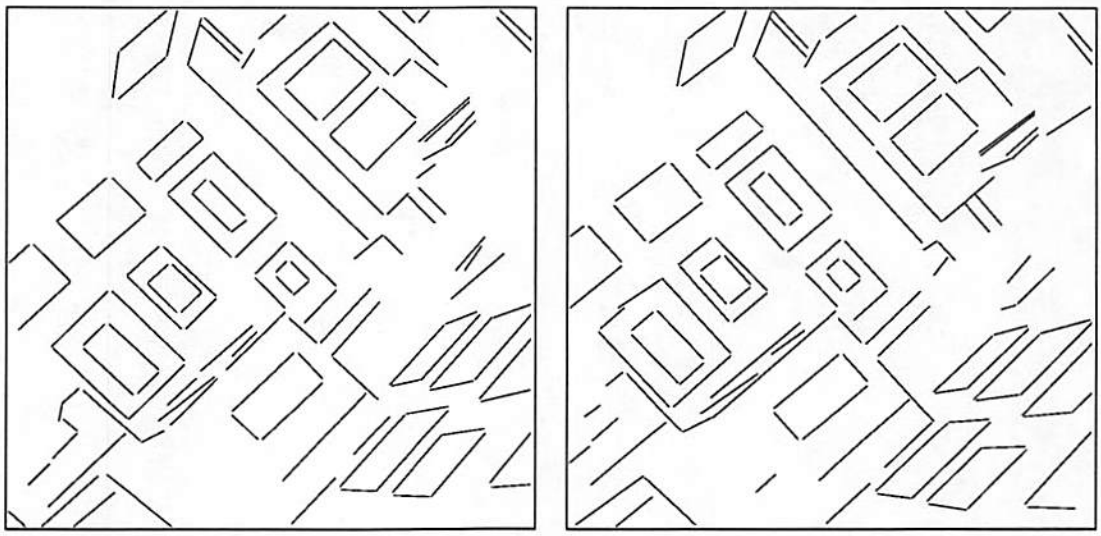


Figure 5.46: Lines extracted from the two views.

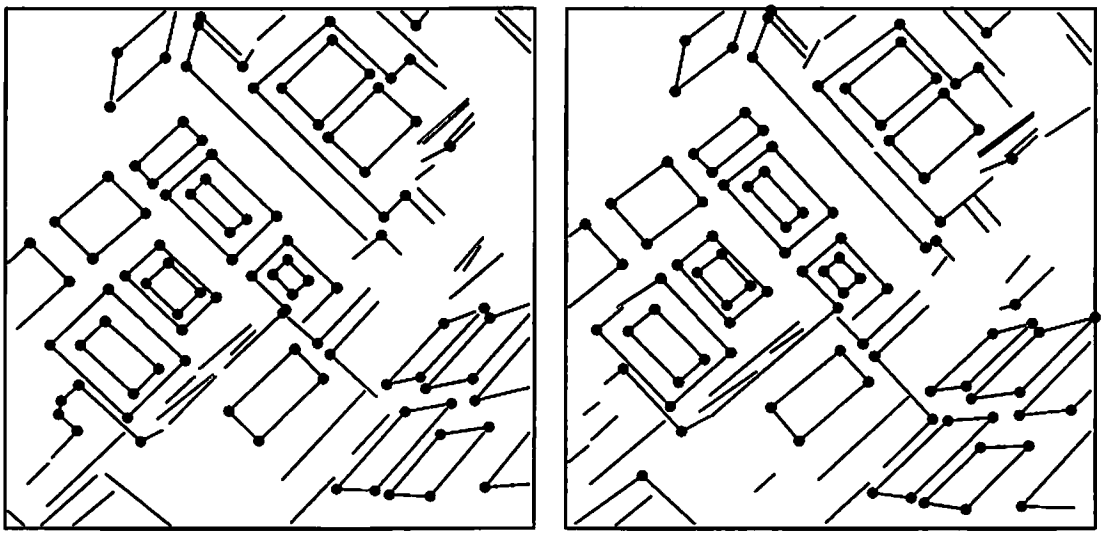


Figure 5.47: Vertices.

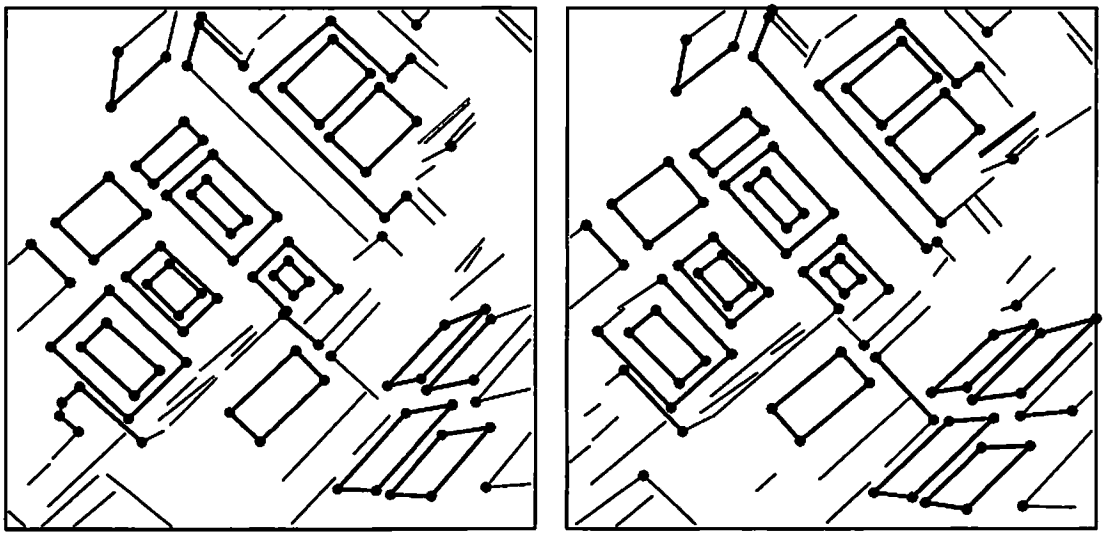


Figure 5.48: Edges (dark lines).

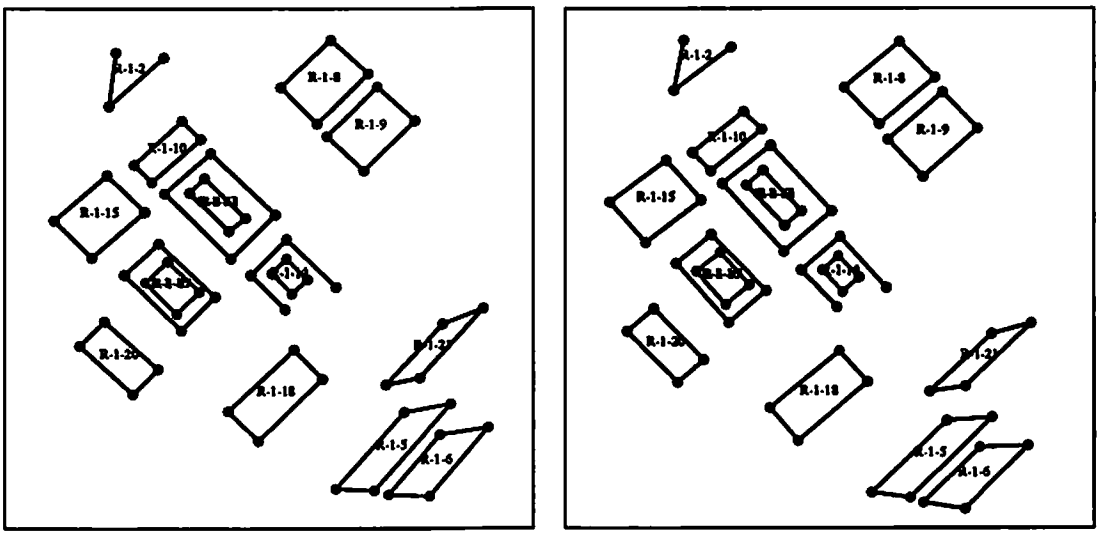


Figure 5.49: After matching at the surface level.

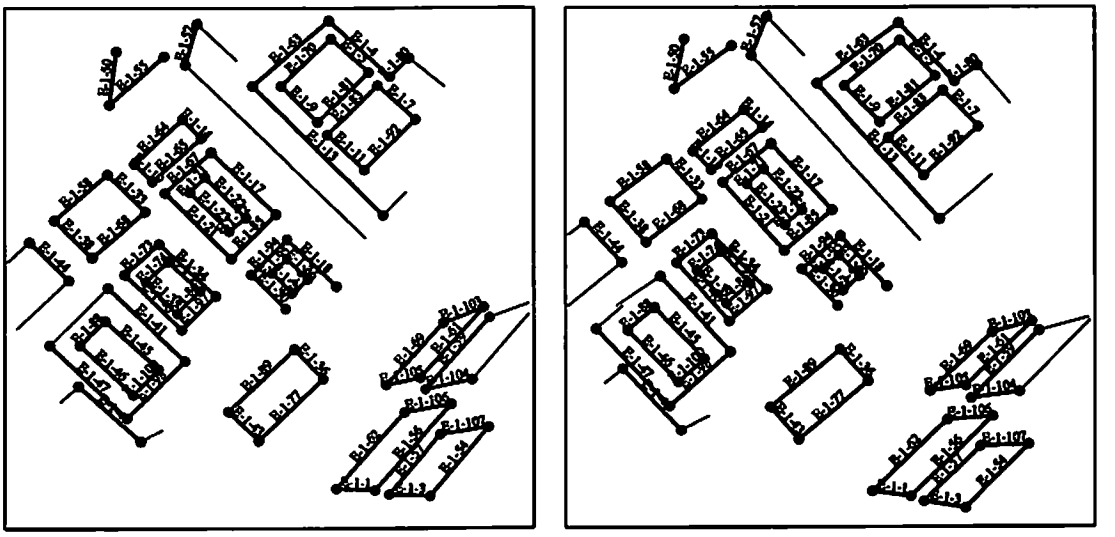


Figure 5.50: After matching at the edge level.

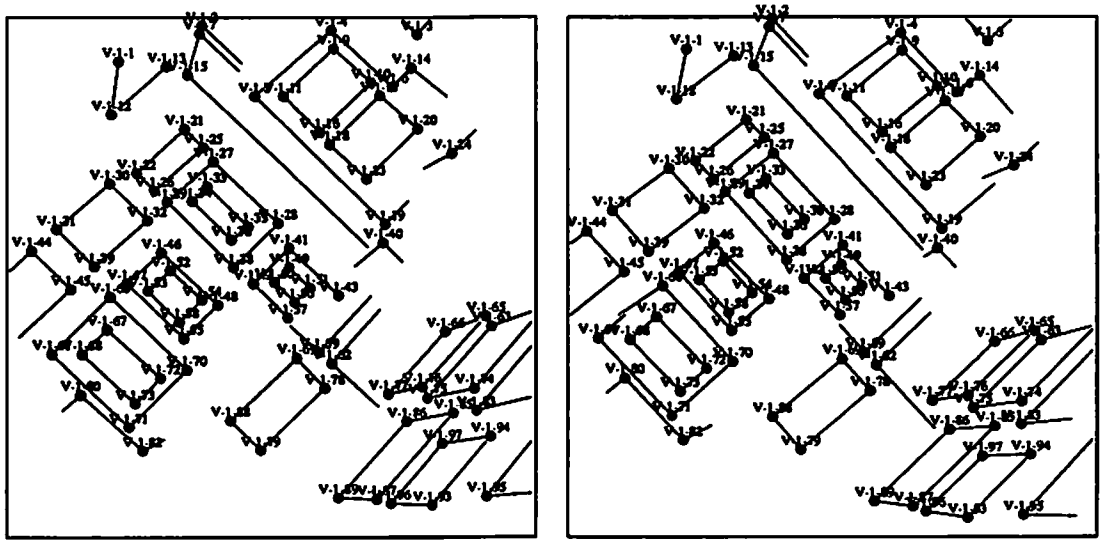


Figure 5.51: After matching at the vertex level.

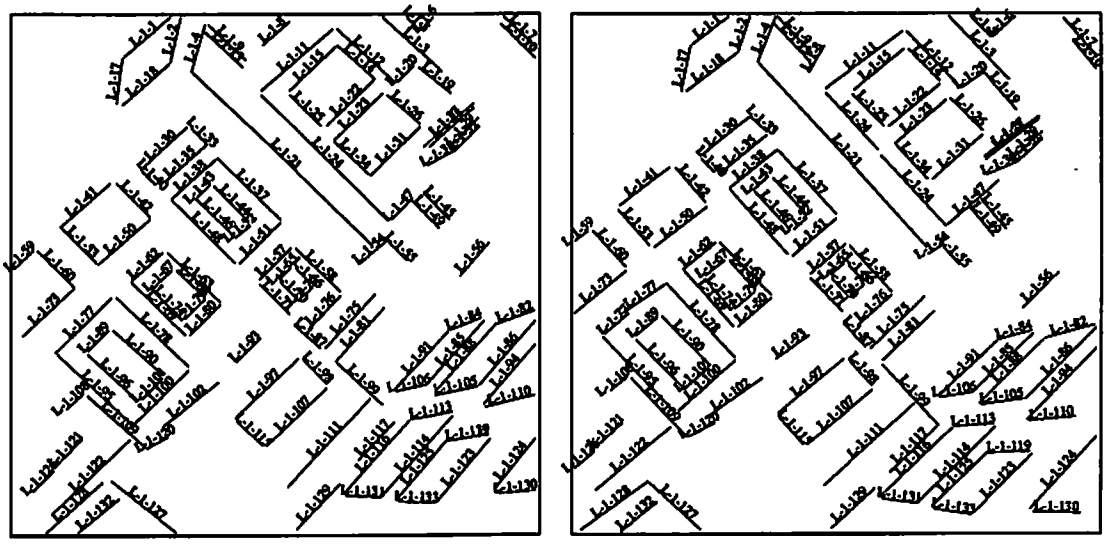


Figure 5.52: Final line matches at the line level.

Figure 5.53: Fifth and ninth frames in the sequence.

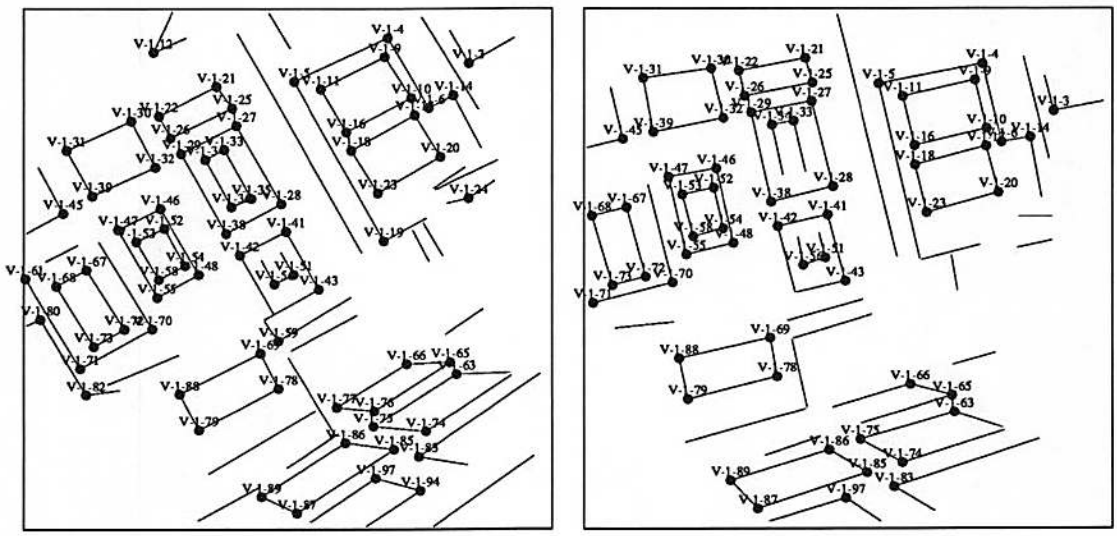


Figure 5.54: Vertices in the fifth and ninth frame tracked from the first frame.

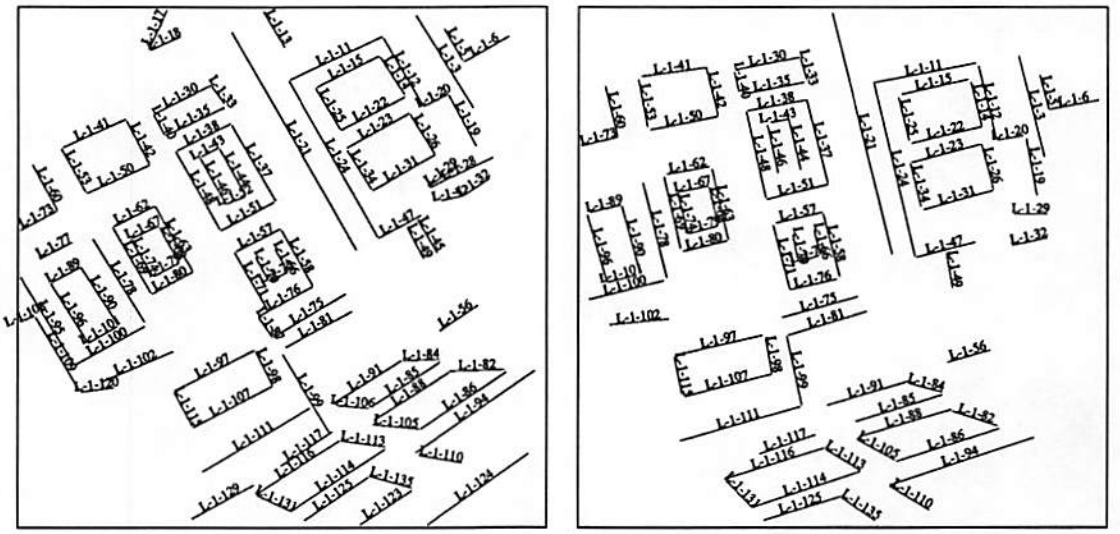


Figure 5.55: Lines in the fifth and ninth frame tracked from the first frame.

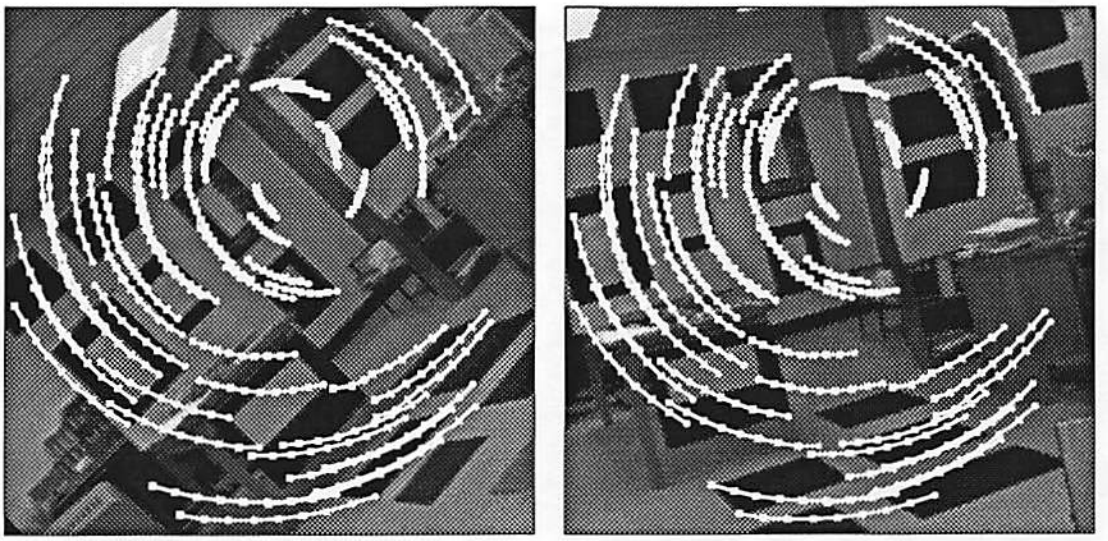


Figure 5.56: Trajectories of vertices originating from first frame and ending in the ninth, superposed on the first and ninth frames.

Chapter 6

Conclusions and Topics for Future Research

6.1 Conclusions

We investigated the issues of representation, search, and matching in computer vision and developed a general framework for addressing each of these issues. Specifically, we made a case for hierarchical scene and model representation, a dynamic search scheme based on an ATMS and a hierarchical matching strategy that uses an ATMS to enforce constraints and derive partial solutions. We then demonstrated three applications that were built based on these framework. For building detection in aerial images, knowledge of the shape of buildings was used to guide a feature grouping strategy. An ATMS assists in dynamic integration of bottom-up and top-down information in a seamless fashion. In stereo and motion correspondence an ATMS assists in enforcing binary, ternary or higher-order constraints and also in deriving partial solutions and confirming them. The

process of confirming partial solutions, leads to a need for belief revision and this is handled by the ATMS.

6.2 Topics for Future Research

The matching techniques presented in this dissertation were for image to image matching. These techniques can be extended to other areas in computer vision where matching with constraints is essential. These include 3D model to 3D data matching [52, 42], image to map correspondence [7], 3D model to 2D data matching [13, 17], etc.

3D model to 3D data matching:

The models can be specified as wireframes. 3D wireframes can be derived from the data by reconstructing depth from the matches derived by the hierarchical stereo matching algorithm described in Chapter 5. Grimson [51] uses only geometric constraints when matching data to models. However he does not group his data into more complex structures. But if one matches the wireframes in a hierarchical fashion one can enforce topological and perceptual constraints as well. This will improve the efficiency and correctness of matching. Also backtracking over an interpretation tree can be avoided by using an ATMS for matching.

3D model to 2D data matching:

In his 3D model to 2D data matching system, Lowe [17] groups perceptually related line segments into larger structures for recognizing objects in 2D images from their 3D Models. Perceptually related segments (like colinear, parallel, etc.) are expected to belong to the same object. But our experience with man-made

scenes shows that different objects in the scene may be arranged so that perceptual relations frequently occur between them. For examples; storage buildings in airports are usually arranged so that their edges are parallel or colinear. So there is a need to augment such groupings with other groupings; for example, groupings based on structural relations. This can further reduce matching complexity. Bodington, Sullivan, and Baker [60] have done a preliminary analysis of the use of an ATMS for matching 3D models to 2D data. Their results suggest that the ATMS is an useful tool, especially if higher order (higher than binary) constraints are involved. More work needs to be done in this area.

Trinocular feature-based stereo:

Hansen, Ayache, and Lustman [118] developed a trinocular stereo system for mobile robot navigation. Three views are used in trinocular stereo, as opposed to two in binocular stereo. In [118], the third image is used to verify the match hypotheses generated by matching the first two. This approach can be used to improve our stereo matching algorithm as well. In this case, feature hierarchies would be derived in each image and feature matches derived from two of them can be confirmed by looking for evidence in the third image.

Recursive motion estimation:

The motion correspondence algorithm described in Chapter 5 matches two images at a time without using estimates of motion from previous frames. One can interleaving matching, estimation, and prediction as in [48, 49, 92]. A uniform motion model can be used along with the first few frames in a batch estimation algorithm to estimate motion. This bootstrap stage of the system will use the algorithm described in Chapter 5. Then the motion estimates can be used to predict the location of features in the next frame. This will simplify matching as

one need to search only in a small area around these locations for the matches. Then these matches can be used to recursively update the motion parameters.

Motion estimation using point and line matches:

Considerable work has been done in the development of algorithms for estimating motion from point matches [77, 78, 79, 80, 81, 82] or line matches [80, 83, 84, 85, 86, 48, 49]. However, the motion correspondence algorithm presented in Chapter 5 gives both point and line matches. Work into using both types of correspondences is still in its infancy [87] and more work needs to be done in this area.

Combining numerical and symbolic approaches to uncertainty:

The ATMS is a symbolic solution to uncertainty reasoning. It cannot deal with degrees of belief in hypotheses like numerical techniques. However, it can reason explicitly about the factors leading to a conflict. Recently, Laskey and Lehner [32] integrated an ATMS with Dempster-Shafer formalism for combining numerical evidence. This was done by associating probabilities with assumptions. A promising area for future research is to use such hybrid schemes to deal with uncertainty during matching, as they combine the advantages of numerical and symbolic approaches.

Image to map matching :

Another interesting area of future research would be image to map matching. Nevatia and Price [7] represent both the image and the map as semantic networks and match them using a sequential algorithm. Price [50] developed a continuous relaxation matching technique for the same task. Features can be arranged hierarchically and a hierarchical matching strategy devised for this problem also.

Multi-sensor fusion :

Traditionally only one sensing modality has been used in computer vision research. For example; in binocular stereo matching discussed in Chapter 5, both images are intensity images obtained using ordinary cameras. The use of multiple imaging modalities can improve interpretation capabilities. The task then would be to merge the information obtained from the different sensors for scene interpretation. Gil, Mitchie, and Aggarwal [119] combine edge maps obtained from range and visible images. One can improve the robustness and efficiency of such merging using a hierarchical representation scheme as discussed in this dissertation. Hebert, Kanade, and Keown [120] combine range and video images into a colored-range image. The scene is segmented into features which have both shape and physical properties. These features are matched to models while enforcing some heuristic constraints based on knowledge of outdoor scenes. This problem can also be tackled within the framework discussed in the dissertation, especially in matching with constraints. Evidential reasoning can be done by using Laskey and Lehner's [32] scheme that merges truth maintenance and Dempster-Shafer theory. Aggarwal and Nandhakumar merge [121] thermal and intensity images. They also make a case for smart search algorithms for establishing correspondence between features obtained from different sensors. Our framework can be used in such tasks too.

Integrating Area Based and Feature Based Approaches:

Feature based stereo matching algorithms (like the one presented in this dissertation) give accurate depth information at the features. This results in a sparse depth map. Area based methods provide dense depth maps but do not give accurate results. The two methods are complementary [67], so we need to integrate

their results to get a dense and accurate depth map. Initial work has been done in this area [65, 122], and this looks like a promising area for future research.

Integrating Different Modules:

One can devise a vision system that has modules for stereo vision, motion detection and object recognition. The issue then would be to develop strategies for integrating the information obtained from these different sources. For example the stereo module would generate 3D wireframes of the scene and the object recognition system would then match 3D models to these wireframe. Examples of such integration already exist. For example, Ayache and Faugeras [123] merge stereo matching with motion estimation. One can also integrate stereo matching with building detection. Currently we use shadows for estimating the heights of roofs. However shadows are not reliable cues because they depend on sunshine. Also we made some simplifying assumptions: shadows were assumed to fall on level ground and not on other buildings nearby. Stereo is a more reliable cue for determining depth.

References

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

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

[3] A. K. Mackworth, Consistency in networks on relations, *Artificial Intelligence*, 8, pp. 99–118, 1977.

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

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

[6] V. Venkateswar and R. Chellappa, Extraction of straight lines in aerial images, In *Proceedings, Fifth European Signal Processing Conference*, pp. 1671–1674, 1990.

[7] R. Nevatia and K. E. Price, Locating structures in aerial images, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–4, pp. 476–484, 1982.

[8] 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.

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

[10] K. M. Andress and A. C. Kak, Evidence accumulation & flow of control in a hierarchical spatial reasoning system, *AI Magazine*, 9(2), pp. 75–94, Summer 1988.

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

[12] 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.

[13] 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.

[14] 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.

[15] H. S. Lim and T. O. Binford, Stereo correspondence: A hierarchical approach, In *Proceedings, DARPA Image Understanding Workshop*, pp. 234–241, Los Angeles, CA, 1987.

[16] H. S. Lim and T. O. Binford, Structural correspondence in stereo vision, In *Proceedings, DARPA Image Understanding Workshop*, pp. 794–808, Cambridge, MA, 1988.

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

[18] R. Mohan and R. Nevatia, Using perceptual organization to extract 3-D structures, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–11, pp. 1121–1139, 1989.

[19] D. M. McKeown, Jr., 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.

[20] C. L. Forgy, The OPS5 user's manual, Technical report, Dept. of Comput. Sci., Carnegie Mellon University, Pittsburgh, PA, 1981.

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

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

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

[24] 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.

[25] J. Doyle, A truth maintenance system, *Artificial Intelligence*, 12, pp. 231–272, 1979.

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

[27] R. K. Bhatnagar and L. N. Kanal, Handling uncertain information: A review of numeric and non-numeric methods, In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pp. 3–26, Elsevier Science Publishers, Amsterdam, The Netherlands, 1988.

[28] J. Pearl, Fusion, propagation, and structuring in belief networks, *Artificial Intelligence*, 29, pp. 241–288, 1986.

[29] G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, Princeton, NJ, 1976.

[30] L. A. Zadeh, Fuzzy sets as a basis for a theory of possibility, *Fuzzy Sets and Systems*, 1, pp. 3–28, 1978.

[31] J. P. Martins, The truth, the whole truth, and nothing but the truth: An indexed bibliography to the literature of Truth Maintenance Systems, *AI Magazine*, 11(5), pp. 7–25, 1991.

[32] K. B. Laskey and P. E. Lehner, Assumptions, beliefs and probabilities, *Artificial Intelligence*, 41, pp. 65–77, 1989/90.

[33] J. Gordon and E. H. Shortliffe, A method for managing evidential reasoning in a hierarchical hypothesis space, *Artificial Intelligence*, 26, pp. 323–357, 1985.

[34] G. Shafer and R. Logan, Implementing Dempster's rule for hierarchical evidence, *Artificial Intelligence*, 33, pp. 271–298, 1987.

[35] 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.

[36] J. B. Bowen and J. E. W. Mayhew, Consistency maintenance in the REV-graph environment, *Image and Vision Computing*, 6, pp. 12–15, 1988.

[37] W. E. L. Grimson and T. Lozano-Perez, Model-based recognition and localization from sparse range or tactile data, *International Journal of Robotics Research*, 3(3), pp. 3–35, 1984.

[38] W. E. L. Grimson and T. Lozano-Perez, Localizing overlapping parts by searching the interpretation tree, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-9, pp. 469–482, 1987.

[39] R. C. Bolles and P. Horaud, 3DPO: A three dimensional part orientation system, *International Journal of Robotics Research*, 5(3), pp. 3–26, Fall 1986.

[40] O. D. Faugeras and M. Hebert, The representation, recognition, and locating of 3-D objects, *International Journal of Robotics Research*, 5(3), pp. 27–52, Fall 1986.

[41] M. Oshima and Y. Shirai, Object recognition using three-dimensional information, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–3, pp. 353–361, 1983.

[42] T.-J. Fan, G. Medioni, and R. Nevatia, Recognizing 3-D objects using surface descriptions, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–11, pp. 1140–1157, 1989.

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

[44] G. Medioni and R. Nevatia, Matching images using linear features, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-6, pp. 675–685, 1984.

[45] G. Medioni and R. Nevatia, Segment-based stereo matching, *Computer Vision, Graphics and Image Processing*, 31, pp. 2–18, 1985.

[46] N. Ayache and B. Faverjon, Efficient registration of stereo images by matching graph descriptions of edge segments, *International Journal of Computer Vision*, 1, pp. 107–131, 1987.

[47] R. Horaud and T. Skordas, Stereo correspondence through feature grouping and maximal cliques, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-11, pp. 1168–1180, 1989.

[48] J. L. Crowley, P. Stelmaszyk, and C. Discours, Measuring image flow by tracking edge-lines, In *Proceedings International Conference on Computer Vision*, pp. 658–664, Tampa, FL, December 1988.

[49] R. Deriche and O. Faugeras, Tracking line segments, In *Proceedings European Conference on Computer Vision*, pp. 259–268, 1990.

[50] K. E. Price, Relaxation matching techniques–a comparison, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–7, pp. 617–623, 1985.

[51] W. E. L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints*, The MIT Press, Cambridge, Massachusetts, 1990.

[52] W. E. L. Grimson, The combinatorics of object recognition in cluttered environments using constrained search, *Artificial Intelligence*, 44, pp. 121–166, 1990.

[53] 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.

[54] A. K. Mackworth and E. C. Freuder, The complexity of some polynomial network consistency algorithms for constraint satisfaction problems, *Artificial Intelligence*, 25, pp. 65–74, 1985.

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

[56] L. G. Shapiro and R. M. Haralick, Structural descriptions and inexact matching, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–3, pp. 504–519, September 1981.

[57] D. H. Ballard and C. M. Brown, *Computer Vision*, Prentice–Hall, Englewood Cliffs, New Jersey, 1982.

[58] R. C. Bolles and R. A. Cain, Recognizing and locating partially visible objects: The local-feature-focus method, *International Journal of Robotics Research*, 1(3), pp. 57–82, 1982.

[59] O. D. Faugeras and K. E. Price, Semantic description of aerial images using stochastic labeling, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI–3, pp. 633–642, November 1981.

[60] R. M. Bodington, G. D. Sullivan, and K. D. Baker, Experiments on the use of the ATMS to label features for object recognition, In *Proceedings European Conference on Computer Vision*, pp. 542–551, 1990.

[61] V. Venkateswar and R. Chellappa, A framework for interpretation of aerial images, In *Proceedings, International Conference on Pattern Recognition, Volume 1*, pp. 204–206, Atlantic City, NJ, June 1990.

[62] V. Venkateswar and R. Chellappa, Hierarchical stereo matching using feature groupings, (submitted for publication).

[63] V. Venkateswar and R. Chellappa, Hierarchical feature based matching for motion correspondence, (submitted for publication).

[64] A. Huertas and R. Nevatia, Detecting buildings in aerial images, *Computer Vision, Graphics and Image Processing*, 41, pp. 131–152, 1988.

[65] D. M. McKeown, Jr., Toward automatic cartographic feature extraction, In L. F. Pau, editor, *Mapping and Spatial Modelling for Navigation*, pp. 149–180, Springer-Verlag, Berlin, 1990.

[66] Y-T Liow and T. Pavlidis, Use of shadows for extracting buildings in aerial images, *Computer Vision, Graphics and Image Processing*, 49, pp. 242–277, 1990.

[67] M. J. Hannah, A system for digital stereo image matching, *Photogrammetric Engineering and Remote Sensing*, 55, pp. 1765–1770, 1989.

[68] D. Marr and T. Poggio, A theory of human stereo vision, *Proceedings Royal Society of London*, B 204, pp. 301–328, 1979.

[69] W. E. L. Grimson, *From Images to Surfaces: A Computational Study of the Human Early Visual System*, The MIT Press, Cambridge, Massachusetts, 1981.

[70] J. E. W. Mayhew and J. P. Frisby, Psychophysical and computational studies towards a theory of human stereopsis, *Artificial Intelligence*, 17, pp. 349–385, 1981.

[71] H. H. Baker and T. O. Binford, A system for automated stereo mapping, In *Proceedings, DARPA Image Understanding Workshop*, pp. 215–222, Palo Alto, CA, 1982.

[72] Y. Ohta and T. Kanade, Stereo by intra- and inter-scanline search using dynamic programming, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-7, pp. 139–154, 1985.

[73] W. Hoff and N. Ahuja, Surfaces from stereo: Integrating feature matching, disparity estimation, and contour detection, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-11, pp. 121–136, 1989.

[74] Y. C. Hsieh, F. Perlant, and D. M. McKeown, Jr., Recovering 3D information from complex aerial imagery, In *Proceedings International Conference on Pattern Recognition, Vol. 1*, pp. 136–146, Atlantic City, NJ, June 1990.

[75] S. T. Barnard and M. A. Fischler, Computational Stereo, *Computing Surveys*, 14, pp. 553–572, 1982.

[76] R. C. K. Chung and R. Nevatia, Use of monocular groupings and occlusion analysis in a hierarchical stereo system, In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition CVPR '91*, Maui, Hawaii, June 1991.

[77] J. W. Roach and J. K. Aggarwal, Determining the movement of objects from a sequence of images, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-2, pp. 554–562, 1980.

[78] H. C. Longuet-Higgins, A computer algorithm for reconstructing a scene from two projections, *Nature*, 293, pp. 133–135, 1981.

[79] R. Y. Tsai and T.S. Huang, Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-6, pp. 13–27, 1984.

[80] O. D. Faugeras, F. Lustman, and G. Toscani, Motion and structure from motion from point and line matches, In *Proceedings International Conference on Computer Vision*, pp. 25–34, London, England, June 1987.

[81] T. J. Broida, S. Chandrasekhar, and R. Chellappa, Recursive 3-D motion estimation from a monocular image sequence, *IEEE Trans. on Aerospace and Electronic Systems*, 26, pp. 639–656, July 1990.

[82] H. Shariat and K. E. Price, Motion estimation with more than two frames, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-12, pp. 417–434, 1990.

[83] M. E. Spetsakis and J. Aloimonos, Closed form solution to the structure from motion problem from line correspondences, In *Proceedings Sixth AAAI, National Conference on Artificial Intelligence*, pp. 738–743, Seattle, WA, July 1987.

[84] Y. Liu and T. S. Huang, A linear algorithm for motion estimation using straight line correspondences, *Computer Vision, Graphics and Image Processing*, 44, pp. 35–57, 1988.

[85] J. Weng, Y. C. Liu, T.S. Huang, and N. Ahuja, Estimating motion/structure from line correspondences: A robust linear algorithm and uniqueness theorems, In *Proceedings IEEE conference on Computer Vision and Pattern Recognition*, pp. 387–392, Ann Arbor, Michigan, June 1988.

[86] J. Weng, T.S. Huang, and N. Ahuja, Estimating motion and structure from line matches: Performance obtained and beyond, In *Proceedings International Conference on Pattern Recognition, Volume 1*, pp. 168–172, Atlantic City, NJ, June 1990.

[87] M. E. Spetsakis and J. Aloimonos, A unified theory of structure from motion, In *Proceedings DARPA Image Understanding Workshop*, pp. 271–283, Pittsburgh, PA, September 1990.

[88] S. T. Barnard and W. B. Thompson, Disparity analysis of images, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-2, pp. 333–340, 1980.

[89] S. Ranade and A. Rosenfeld, Point pattern matching by relaxation, *Pattern Recognition*, 12, pp. 269–275, 1980.

[90] J-Q Fang and T. S. Huang, Some experiments on estimating the 3-D motion parameters of a rigid body from two consecutive image frames, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-6, pp. 545–554, 1984.

[91] I. K. Sethi and R. Jain, Finding trajectories of feature points in a monocular image sequence, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-9, pp. 56–73, 1987.

[92] S. Chandrasekhar and R. Chellappa, Passive navigation in a partially known environment, (submitted for publication).

[93] R. O. Duda and P. E. Hart, Use of the Hough transformation to detect lines and curves in pictures, *Communications of the ACM*, 15, pp. 11–15, 1972.

[94] S. D. Shapiro, Feature space transforms for curve detection, *Pattern Recognition*, 10, pp. 129–143, 1978.

[95] S. D. Shapiro, Properties of transforms for the detection of curves in noisy pictures, *Computer Graphics and Image Processing*, 8, pp. 219–236, 1978.

[96] W. E. L. Grimson and D.P. Huttenlocher, On the sensitivity of the Hough transform for object recognition, *IEEE Trans. on Patt. Anal. and Mach. Intell.*, PAMI-12, pp. 255–274, 1990.

[97] D. H. Ballard, Generalizing the Hough transform to detect arbitrary shapes, *Pattern Recognition*, 13, pp. 111–122, 1981.

[98] J. Illingworth and J. Kittler, A survey of the Hough transform, *Computer Vision, Graphics and Image Processing*, 44, pp. 87–116, 1988.

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

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

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

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

[103] R. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Computation*, 1, pp. 146–160, 1972.

[104] V. Venkateswar and R. Chellappa, Intelligent interpretation of aerial images, Technical Report USC-SIPI-137, University of Southern California, March 1989.

[105] S. A. Shafer and T. Kanade, Using shadows in finding surface orientations, *Computer Vision, Graphics and Image Processing*, 22, pp. 145–176, 1983.

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

[107] R. B. Irvin and D. M. McKeown, Jr., Methods for exploiting the relationship between buildings and their shadows in aerial imagery, Technical Report CMU–CS–88–200, Carnegie Mellon University, December 1988.

[108] D. G. Lowe, *Perceptual organization and visual recognition*, Kluwer Academic, Boston, MA, 1985.

[109] D. E. Knuth, *The Art of Computer Programming, Vol. 3*, Addison Wesley, Reading, Massachusetts, 1973.

[110] H. J. Levesque and R. J. Brachman, A fundamental tradeoff in knowledge representation and reasoning, In R. J. Brachman and H. J. Levesque, editors, *Readings in Knowledge Representation*, pp. 42–70, Morgan Kaufman Publishers, Inc., Los Altos, CA, 1985.

[111] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence*, Addison Wesley, Reading, Massachusetts, 1985.

[112] R. Davis and J. King, An overview of production systems, In E. Elcock and D. Michie, editors, *Machine Intelligence 8: Machine Representations of Knowledge*, pp. 300–332, Wiley, New York, 1977.

[113] N. Ayache and C. Hansen, Rectification of images for binocular and trinocular stereo vision, In *Proceedings International Conference on Pattern Recognition*, Rome, Italy, 1988.

[114] R. D. Arnold and T. O. Binford, Geometric constraints in stereo vision, In *Proceedings, SPIE Vol.238–Image Processing for Missile Guidance*, pp. 281–292, 1980.

[115] 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.

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

[117] H. S. Lim and T. O. Binford, Curved surface reconstruction using stereo correspondence, In *Proceedings, DARPA Image Understanding Workshop*, pp. 809–819, Cambridge, MA, 1988.

[118] C. Hansen, N. Ayache, and F. Lustman, High speed trinocular stereo for mobile-robot navigation, In J. T. Tou and J. G. Balchen, editors, *Highly Redundant Sensing Systems*, pp. 127–146, Springer Verlag, New York, 1990.

[119] B. Gil, A. Mitchie, and J. K. Aggarwal, Experiments in combining intensity and range edge maps, *Computer Vision, Graphics and Image Processing*, 21, pp. 395–411, 1983.

[120] M. Hebert, T. Kanade, and I. Kweon, 3-D vision techniques for autonomous vehicles, In R. C. Jain and A. K. Jain, editors, *Analysis and Interpretation of Range Images*, pp. 273–337, Springer-Verlag, New York, 1990.

[121] J. K. Aggarwal and N. Nandhakumar, Multisensor fusion for automatic scene interpretation, In R. C. Jain and A. K. Jain, editors, *Analysis and Interpretation of Range Images*, pp. 339–361, Springer-Verlag, New York, 1990.

[122] S. D. Cochran, *Surface Description from Binocular Stereo*, PhD thesis, University of Southern California, November 1990.

[123] N. Ayache and O. D. Faugeras, Building, registrating, and fusing noisy visual maps, In *Proceedings International Conference on Computer Vision*, pp. 73–82, London, England, June 1987.