

**INCREMENTAL CONSTRUCTION
OF 3-D MODELS OF A SCENE
FROM SEQUENTIALLY PLANNED VIEWS**

by

Shun-en Xie

Diploma, University of Science and Technology of China, 1967

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Ph.D.

in the School

of

Computing Science

© Shun-en Xie 1986

SIMON FRASER UNIVERSITY

December 1986

All rights reserved. This thesis may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.

Name: Shun-en Xie

Degree: Ph.D.

Title of Thesis: INCREMENTAL CONSTRUCTION OF 3-D MODELS OF A
SCENE FROM SEQUENTIALLY PLANNED VIEWS

Examining Committee:

Chairperson: Dr. James P. Delgrande

Dr. Thomas W. Calvert
School of Computing Science
Senior Supervisor

Dr. Nick J. Cercone
School of Computing Science
Internal External Examiner

Dr. Brian V. Funt
School of Computing Science

Dr. Andrew. K. C. Wong
Dept. of Systems Design Engineering
University of Waterloo
External Examiner

Dr. Binay K. Bhattacharya
School of Computing Science

12 December 1986
Date Approved

Dr. B. T. (Tad) McGeer
Dept. of Engineering Science

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/~~Project/Extended Essay~~

Incremental Construction of 3-D Models of a Scene
from Sequentially Planned Views

Author: _____

(signature)

SHUN-EN XIE

(name)

Dec. 16, 1986

(date)

Acknowledgements

Acknowledgements

This section gives me an opportunity to express my gratitude to a number of people whose support and encouragement made this research and thesis possible.

At the top of this list is my supervisor, Dr. Thomas W. Calvert. Not only did he get me started and provide much support but also provided much valuable advice. When I met obstacles, he always stood behind me and encouraged me.

Dr. Binay K. Bhattacharya had a very strong influence on the research which I did on the viewpoint planning, especially the algorithms in the 2-D case. I would like to thank Dr. Brian V. Funt and Dr. Tad McGeer for their helpful discussions and suggestions. I am also grateful to the staff of the LCCR lab and the School of Computing Science for providing various aids.

On the financial side, this dissertation is supported mainly by the C.D. Nelson Memorial Scholarship and in part by grants from the NSERC of Canada. My aunt, Dr. Y.S. Hsieh, and uncle, Dr. S. Hsu, were also generous contributors.

Finally, special thanks go to my wife and parents for their patience and cheerful encouragement.

ABSTRACT

In the future intelligent robots will be called upon to play many important roles in business, industrial and domestic situations. Whether in an office, a warehouse or a home, the mobile robot will have to work in a cluttered environment: although the basic layout of the environment may be known in advance, the nature and placement of objects within the environment will generally be unknown. Thus the intelligent mobile robot must be able to sense its environment with a vision system and it must be able to analyse multiple views to construct 3-d models of the objects it encounters. Since this analysis results in a heavy computational load, it is important to minimize the number of views and to use a planner to dynamically select a minimal set of vantage viewpoints.

This thesis discusses an approach to this general problem and describes a prototype system for a mobile intelligent robot which can construct 3-d models from planned sequential views. The principal components of this system are:

1. decomposition of a framed view into its components and construction of partial 3-D descriptions of the view,
2. matching of the known environment to partial 3-D descriptions of the view,
3. matching of partial descriptions of bodies derived from the current view with partial models constructed from previous views,
4. identification of new information in the current view and use of the information to update the models.

5. identification of unknown parts of partially constructed body models so that further viewpoints can be planned.
6. construction of a partial map of the scene and updating with each successive view.
7. selection of new viewpoints to maximize the information returned by a planner.
8. use of an expert system to convert the constructed boundary representations of the bodies to a new Constructive Solid Geometry-Extended Enhanced Spherical Image (CSG-EESI) representation to facilitate the recovery of structural information.

Although the complete prototype system has not been implemented, its key components have been implemented and tested.

Table of Contents

Acknowledgements	iii
ABSTRACT	iv
Table of Contents	vi
List of Tables	ix
List of Figures	x
1. Introduction	1
1.1. Problem Specification and Thesis Outline	1
1.2. A Sketch of the Prototype System	8
2. Review of the Literature	12
2.1. General Intelligent Vision Systems	12
2.2. Review of Work on Interpreting Line Drawings from a Single View	15
2.3. Review of Work on Analysing a Scene from Multiple Views	20
2.4. Representation of Solids and the World	25
2.4.1. Model of Solid Bodies	25
2.4.2. Model of the World	31
2.5. Review of Work on Planning	34
2.5.1. Methodology of Planner	34
2.5.2. Trajectory Planning	37
2.5.3. Viewpoint Planning	43
2.6. Expert Systems	46
3. Incremental Construction of 3-D Models from a Sequence of Framed Views	50
3.1. Introduction	50
3.2. Labeling Scheme	53
3.3. Decomposition	56
3.4. Matching the Environment Model	58
3.5. Matching Partial Body Models	60
3.6. Model Updating	62
3.7. Identification of Unknown Parts	63
3.8. Experiment	64
3.9. Chapter Summary	66

4. Construction of the Environment Map	69
4.1. Introduction	69
4.2. Construction of a map for a scene consisting of known bodies	71
4.3. Construction of a map for a scene consisting of partially known polyhedron bodies	75
4.4. Construction of a map for a scene consisting of partially known curved bodies	76
4.5. An Example	80
5. Planning Viewpoints and the Navigation Route of a Patrol Robot in a Known 2-D Environment	82
5.1. Introduction	82
5.2. Planning Viewpoints in 2-D	84
5.2.1. Partition Scheme	85
5.2.2. Covering Scheme	87
5.3. Planning the Navigation Route	90
5.4. Chapter Summary	92
6. Planning Views for the Incremental Construction of Body Models	95
6.1. Introduction	95
6.2. General Approach	96
6.3. Planning Viewpoints in 2-D for Unknown Objects	97
6.4. Planning Viewpoints for 3-D Local Ambiguities	100
6.5. Chapter Summary	103
7. CSG-EESI: A New Solid Representation Scheme and a Conversion Expert System	104
7.1. Introduction	104
7.2. The CSG-EESI Representation	107
7.3. Conversion from BR to CSG-EESI	116
7.4. The Conversion Expert System	118
7.5. Experiments	124
7.6. Discussion and Chapter Summary	124
8. The Organization and Function of the Prototype System	130
9. Conclusions	137
Appendix A. Format of Input and Output Data	141
Appendix B. Rules for Matching Multiple Views	146

Appendix C. Junction Labeling Scheme	152
Appendix D. Knowledge for Labeling Junctions and Their Related Lines	159
Appendix E. The Dictionary of Junction Families	165
Appendix F. The Mathematical Description of a Quadric Surface or a Curve	167
Appendix G. Some Mathematical Formulas of Spherical Projection	169
Appendix H. Rules for Solving Local Ambiguities	173
Appendix I. Knowledge for the CSG-EESI Conversion	175
Appendix J. The Input and Output Forms of a Test Body for the CSG-EESI Conversion System	187
References	191

List of Tables

Table 7-1:	Vector components in the EESI scheme.	109
Table 7-2:	The location of the origin of the local coordinate system for a primitive body.	111

List of Figures

Figure 1-1:	A typical office scene which has been used in this research.	4
Figure 1-2:	The schematic sketch of the prototype system.	9
Figure 3-1:	A schematic sketch of the system for matching and model construction.	51
Figure 3-2:	The types of junctions.	54
Figure 3-3:	An example scene labeled by the labeling scheme.	55
Figure 3-4:	The two synthesized views which have been successfully analyzed by the system for matching and model construction.	67
Figure 4-1:	The construction of the silhouette of a difference body when the silhouette of the secondary part totally overlaps the silhouette of the primary part.	73
Figure 4-2:	The construction of the silhouette of a difference body when a part of peripheral silhouette of the primary part is overlapped by the silhouette of the second part.	73
Figure 4-3:	A counter-example for the difference operation affecting projections when the primary part is not convex.	74
Figure 4-4:	The partial map of an example scene constructed from Viewpoint 1.	81
Figure 4-5:	The partial map of an example scene constructed after Viewpoint 2.	81
Figure 4-6:	The partial map of an example scene constructed after Viewpoint 3.	81
Figure 4-7:	The partial map of an example scene constructed after Viewpoint 4.	81
Figure 4-8:	The partial map of an example scene constructed after Viewpoint 5.	81
Figure 4-9:	The partial map of an example scene constructed after Viewpoint 6.	81
Figure 5-1:	Scene 5-1: A test example for a cluttered scene.	86
Figure 5-2:	Scene 5-2: A more realistic scene.	86
Figure 5-3:	The selected viewpoints for Scene 5-1 using the partition scheme.	93

Figure 5-4:	The selected viewpoints for Scene 5-1 using the covering scheme.	93
Figure 5-5:	The star-shaped polygonal partition and the graph of passways for Scene 5-1.	93
Figure 5-6:	The convex polygonal partition and the graph of passways for Scene 5-1.	93
Figure 5-7:	The reordered viewpoints and corresponding robot navigation route for Scene 5-1 using the partition scheme.	93
Figure 5-8:	The reordered viewpoints and corresponding robot navigation route for Scene 5-1 using the covering scheme.	93
Figure 5-9:	The selected viewpoints for Scene 5-2 using the partition scheme.	94
Figure 5-10:	The selected viewpoints for Scene 5-2 using the covering scheme.	94
Figure 5-11:	The star-shaped polygonal partition and the graph of passways for Scene 5-2.	94
Figure 5-12:	The convex polygonal partition and the graph of passways for Scene 5-2.	94
Figure 5-13:	The reordered viewpoints and corresponding robot navigation route for Scene 5-2 using the partition scheme.	94
Figure 5-14:	The reordered viewpoints and corresponding robot navigation route for Scene 5-2 using the covering scheme.	94
Figure 6-1:	A solution for Scene 5-1 where no prior knowledge is assumed for the objects.	101
Figure 6-2:	A solution for Scene 5-2 where no prior knowledge is assumed for the objects.	101
Figure 6-3:	A solution for Scene 5-2 with an improved exploration route for the robot.	102
Figure 7-1:	Two examples of one kind of formal representation for the EESI model.	112
Figure 7-2:	The schematic sketch of the conversion expert system.	118
Figure 7-3:	An example of a decomposition made by the BR to CSG-EESI Conversion Expert System.	122
Figure 7-4:	An example of the local difference which can result between an "exact" decomposition and that achieved by the expert conversion system.	123
Figure 7-5:	The test group with various types of faces and connectivities compares with its decomposition results. The CPU times are shown under each decomposition result.	125
Figure 7-6:	Another test group with various types of structure compares with its decomposition results. The CPU times are shown under each decomposition result.	126

Figure 7-7:	A proposed system for automatic model formation.	128
Figure 8-1:	The major modules and the organization of the prototype vision system for a mobile robot.	131
Figure 8-2:	A schematic sketch of the control system for a mobile robot.	133
Figure G-1:	The spherical coordinate system based on the robot's eye.	169

Chapter 1

Introduction

1.1. Problem Specification and Thesis Outline

In the future intelligent robots will be called upon to play many important roles in industrial, business and domestic situations. As Nitzan [65] pointed out, a flexible, intelligent robot is regarded as a general purpose machine system that may include effectors, sensors, computers, and auxiliary equipment and which like a human, can perform a variety of tasks under unpredictable conditions. Flexibility means the ability to perform a class of different tasks; intelligence means the ability of a machine system to perceive conditions that may not have been known *a priori*, decide what actions should be performed, and plan these actions accordingly. If these robots are to work in complex environments, it will be necessary to develop knowledge-based sensory systems. In simple situations, the robot vision system can have built-in models of both the environment and all objects within it; this allows a relatively simple recognition process. In more realistic situations, however, although the geometry of the surrounding environment may be known (i.e. the dimensions of the room, warehouse, etc, in which the robot operates), the type and position of the objects in the environment will generally be unknown. Thus knowledge of the structure and placement of these objects must be learned. To do this the mobile robot must first construct 3-

dimensional models for the objects it encounters. It should then be possible to classify these objects by comparing their structural properties with those of generally known classes of objects such as benches, chairs, tables, etc.

Such a robot should carry out the following tasks:

1. Movement. The robot should be able to move through its environment from location A to location B. This involves finding its position within the environment and choosing a path to move to a new position while avoiding fixed objects and passive movable objects.
2. Location, Recognition and Manipulation. The robot should be able to determine the position and orientation of well defined objects within the environment. Having found an object, the robot should be able to manipulate and move it.

To overcome the limitations of today's "muscle-only" robots, it is necessary to provide them with such intelligent robot functions as attention, planning, sensing, learning and knowledge rectification and to elegantly combine these functions together.

In the 1960's and 1970's, visual image processing and analysis of static scenes received considerable attention [8]. But, in analyzing a single framed view of part of a large scene, there are certain problems which generally stand in the way of constructing the 3-D body models; these include: partial features, self-occlusion, occlusion, accidental alignment, special alignment, and undetermined geometric parameters.

In general, vision is not a single action; it is a sequential recursive and cyclical

process of alternating information gathering and decision making. Thus, an approach to understanding a scene from image sequences by incrementally constructing body models seems promising. However, even today, the information processing load involved in analyzing a sequence of images presents a serious technical problem. Dynamic selection of a minimal set of vantage viewpoints and effective selection of only the necessary information will be essential if the burden of computation is to be lightened. Fortunately, a mobile robot, by its nature, offers a good foundation for gathering information from different points of view. Thus combining a vision system with a planner, so that a scene can be analyzed from planned multiple views, is both natural and necessary.

In the natural world, many birds and some animals have a long flexible neck. The neck plays an important auxiliary function in their vision activity. Even a human often uses his neck to adjust the azimuth of his view. Thus we assume that the mobile robot considered has a long neck, which can stretch up and down, and an eye (camera) which is on top of the neck and which can turn around to nearly any direction. Though the visual angle of the robot's eye is narrow, its field of vision is wide. From the view of ecological optics, spherical projection models the true physical appearance of an observed scene more accurately than planar projection, especially in a large scene. Thus, it is natural to assume that the advanced robot has a sensor which can sense the spherical projection of a scene. In terms of a possible realization of spherical projection, a fish-eye lens may be used as an imaging device.

The goal of this thesis is to develop the methodologies and strategies for such a

robot and to design a prototype system which can construct 3-dimensional models of the objects within a practical domestic, business or industrial environment based on sequential planned views. This kind of 3-D model provides both structural and geometric information for further body classification. Although several general vision systems have been developed or are being developed (See Chapter 2), most of them concentrate on low-level vision. Here we are concerned with high-level vision and the combination of vision with planning. We do not intend to build a practical vision system at the present time, since there is no single existing system which can successfully deal with low-level processing and extract satisfactory 2 1/2 D sketches. We hope that the methodologies and prototype system developed in this thesis will not only accelerate the development of high-level vision processing for intelligent robots, but will also contribute to the synthesis of low-level vision with high-level vision.

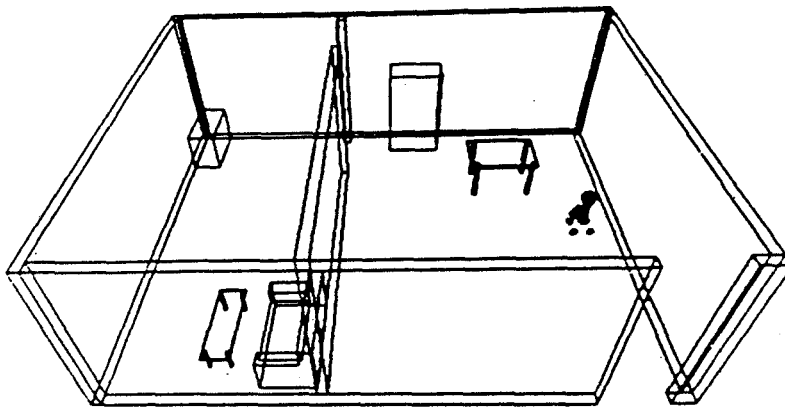


Figure 1-1: A typical office scene which has been used in this research.

A typical scene which has been used in this research to test the prototype

system is shown in Figure 1-1. The scene has been created by an interactive graphics system which has been developed to simulate the movement and sensing of a mobile robot on an IRIS workstation in programming language C.

To limit the scope of the immediate research problem the following assumptions have been made:

1. The bodies in the environment are static, rigid, weakly externally visible¹, and have vertices formed by at most three surfaces. Edges are formed by two surfaces, which can be planar, conical, cylindrical or spherical. Certain complex bodies which consist of concave surfaces will be excluded.
2. There is a preprocessor which deals with early and intermediate vision processing of the visual data. The output of the preprocessor is equivalent to a complete 2 1/2 D sketch. The categories of facets and lines, the orientations of planes and the rough depths of junctions have already been extracted from the 2 1/2 D sketch. This is facilitated if (a) the only lights in the environment are a point source and a diffuse source, and (b) a pinhole spherical camera model is used to acquire the images.
3. The shape and dimensions of the robot environment are given.
4. The robot is only allowed to move around bodies in order to view them (i.e. it cannot go over them or underneath them).

This research has concentrated on the following principal elements:

1. decomposition of a framed view into its components and construction of partial 3-D descriptions of the view.

¹A body is weakly externally visible iff all of its boundary points are weakly visible from a sphere enclosing the body.

2. matching of the known environment to the partial 3-D descriptions of the view.
3. matching of partial descriptions of bodies derived from the current view with partial models constructed from previous views.
4. identification of new information in the current view and use of the information to update the models.
5. identification of unknown parts of partially constructed body models so that further viewpoints can be planned.
6. construction of a partial map of the scene and updating it with each successive view.
7. selection of new viewpoints to maximize the information returned by a planner.
8. use of an expert system to convert the constructed boundary representations of the bodies to a new Constructive Solid Geometry-Extended Enhanced Spherical Image (CSG-EESI) representation to facilitate the recovery of structural information.

More precisely, the input of the proposed high-level vision processing system is a set of faces with their bounding lines and junctions encoded by 2-D spherical coordinates and depths measured from the viewpoint concerned. The directions of face normals are also provided. There are tolerance errors for the depths of vertices, the directions of face normals and the positions of viewpoints. The principal features of the environment will be part of the system knowledge base and built into what will be designated the **Long Term Memory(LTM)**. The detailed input and output formats are described in Appendix A.

For each view, the partial 3-D descriptions of bodies are derived by labeling and

segmenting the image. After the environment model has been matched, the partial 3-D descriptions of the first view will be used as the initially constructed partial models of the bodies in the scene. These models will be refined by choosing further views sequentially until the model is complete and precise enough to carry out certain tasks. This part of research work is described in Chapter 3.

Based on the partially constructed models of the bodies and the built-in model of the environment, a 2-D map of the robot environment can be constructed (described in Chapter 4). If partially constructed models remain incomplete after a sequence of views, then they are analyzed by the viewpoint planning system (described in Chapter 6); using this system, new views can be chosen to resolve the ambiguities based on the built-in knowledge and the accumulated information from previous views. In any realistic situation, we would expect that the task assigned to the robot would also provide input to the planning system so that a decision could be made to ignore incomplete objects that are irrelevant to the current task.

The body models (either partial or complete) which are constructed from the multiple views have Boundary Representation (BR)-like representations. Once a complete model has been constructed, a rule-based conversion system (described in Chapter 7) is used to transform the BR representation into our new CSG-EESI representation. Thus it provides both structural and geometric information for the bodies and higher level 3-D models can be more easily derived. The CSG-EESI representation facilitates object classification by allowing comparison of unknown object structures with those for prototype objects which might be expected to be in the environment.

The final result of analyzing a scene by the system is an aerial map of the scene and the 3-D CSG-EESI models for the bodies within the environment. The map indicates the locations of the bodies and the relationships between them. Also the trajectory of the robot movement is recorded and displayed on the map. Methods have been developed (described in Chapter 5 and 6) to plan optional routes through an environment for which the map is either complete or partially known.

The structure of the prototype system is described in the next section; and the organization and function of the prototype system is described in Chapter 8. Although the basis for each component of the system is described in appropriate detail, no attempt has been made to implement a complete prototype system. However, certain key components have been implemented and tested.

1.2. A Sketch of the Prototype System

Computer vision and planning are both examples of complex problem solving. Computer vision deals with various kinds of information and involves the control of different information processors. In analogy to the structure of the human brain, where there are many information processing centers with connections among them, a hierarchical expert system has been devised for the robot vision system. This consists of: a super-expert system called super-controller, a collection of sub-expert and micro-expert systems, a Short Term Memory (STM), and a Long Term Memory (LTM). Figure 1-2 is a schematic of the prototype system.

The principal motivation for this approach is to provide an extensible and flexible

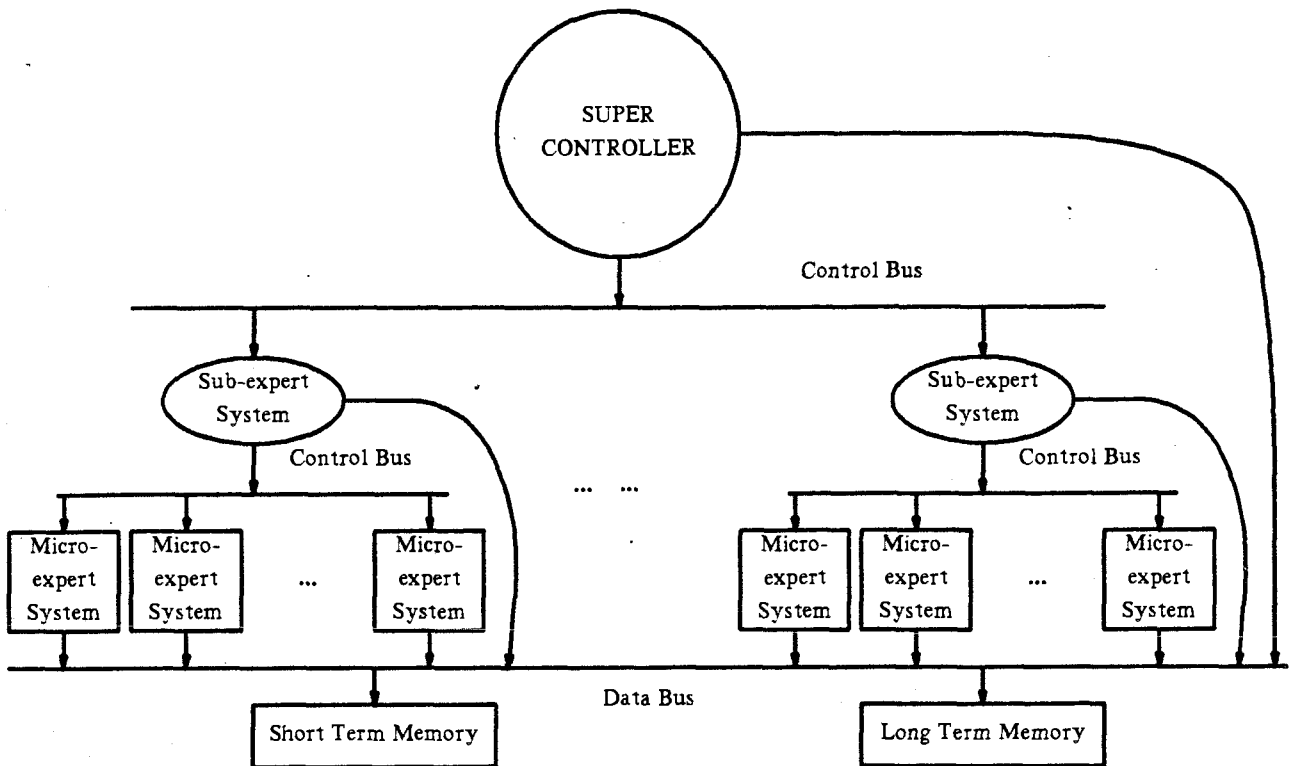


Figure 1-2: The schematic sketch of the prototype system.

expert system architecture for solving a complex problem. A complex problem consists of many tasks, each of which may consist of sub-tasks. Experts or expert committees will be assigned to deal with the tasks or sub-tasks of the problem individually. To control and to coordinate the activities of the experts and expert committees a head-expert is needed.

In the prototype system, Short Term Memory is the temporary database. It is a collection of individual micro-databases which contain the following information:

1. the data for each input image.
2. the data for the 3-D models which have been constructed.
3. the data for the constructed map.
4. the data for the coordinate systems including a fixed orthogonal coordinate system and the spherical coordinate systems related to each viewpoint.

At any stage of an analysis process, only the necessary micro databases of the STM will be dynamically loaded into the main memory. Long Term Memory is the permanent knowledge/data base and contains the model of the scene environment. The information in the LTM is unchangeable during the period of analysis, but the STM is updated and revised by the analysis processors. The super-controller acts as the highest level planner. In it, there is a stateword which is a control level micro-database and which retains some state information that indicates the new contributions and the resulting status of all sub-expert systems. Each sub-expert system can communicate with the super-controller, Short Term Memory, Long Term Memory and its own micro-expert systems; and each micro-expert system can communicate with its parent sub-expert system, Short Term Memory, Long Term Memory and with the super-controller through its parent sub-expert system. Thus the sub-expert and micro-expert systems can communicate and cooperate with each other. Each sub-expert system and micro-expert system can be viewed as an analysis processor which deals with certain tasks. The sub-expert systems usually act as planners at the lower levels. The super-controller may also be viewed as a controlled production system. According to the current content of the stateword, the super-controller forms a current task, and

corresponding sub-expert and micro-expert systems are invoked. According to functions, the sub-expert and micro-expert systems are classified and organized in a natural way.

In the prototype system which has been developed, different knowledge representations (e. g., production rule, general pattern-invoked program and first order predicate logic) and different control strategies (e.g., data-driven search, model-driven search, propagation search and backtracking) are used in different expert systems for different analysis tasks. To a certain degree, this hierarchical expert system will function as a heterarchically structured system. This prototype system has the properties of extensibility and modularity of both the knowledge and the control mechanisms. Thus it can be easily expanded to include more analysis processors.

Chapter 2

Review of the Literature

2.1. General Intelligent Vision Systems

Intelligent robots are general-purpose machine systems performing a variety of tasks under conditions that may not be known *a priori*. Thus they need to be equipped with a general vision system. There is active research in this field, but it will take a considerable time to achieve the expected goal: a general intelligent vision system for real-time robots. Shirai [88] has surveyed previous research work and has pointed out that in order to realize flexible manufacturing systems, there is a need for more advanced robots that can adapt to new situations easily. One of the most important techniques required by such flexible robots is the ability to understand the environment through visual information. Although it is impossible, in principle, to capture three-dimensional information from a monocular image, if something is known about the scene, a monocular image or multiple monocular images (viewed from different directions) may often provide three-dimensional information. In general, the shape of objects can be inferred by surface orientations. Methods of shape recovery depend on the constraints imposed on the scene and the available input information. According to the constraints, the methods have such names as shape from shading or shape from texture.

Lavin [47] pointed out that the three-dimensional structure of the environment,

together with the laws of optics and photometry, determine the structure of the two-dimensional visual field. Given certain assumptions, it is often possible to reverse the mapping, recovering some of the three-dimensional information from the two-dimensional image. He constructed a program DYNAPU which uses a series of snapshots of hilly terrain (line drawings) to construct a map of the terrain and discover the positions of the camera.

Lee and Fu [50] proposed a computer vision system which combines a Bottom-up approach with a Top-down approach in an interactive manner. The input of the system is a single-view, gray-level image. The bottom-up analysis is first used to extract the primitives and relations and construct a precursory object description, which is then used to retrieve proper object models from the associative memory network. The top-down approach is used to verify the models. If all candidate models are rejected, the top-down process asks to re-group the extracted primitives and relations for another try.

McCain [60] described a material handling robot arm developed at the National Bureau of Standards. This robot arm has been equipped with a hierarchical robot control system, a finger force sensor, a 3-D vision system, a watchdog safety system and an instrumented, servo-controlled gripper. Its vision system is mounted on the end of its arm and consists of a camera and two light projectors: dual line flash and flood flash. In its vision system, when the dual line flash is fired, from the image produced it is possible to compute the range and the pitch and yaw orientations of any simple shaped surface from the image produced. From the flood image the vertical and horizontal positions of the surface can be

computed and with additional computation (using two points on the surface) the roll orientation of the surface can be calculated. Thus all six degrees of freedom that define the position and orientation of any object relative to the robot gripper are determined. The image data is further processed for specific part identification. It is processed into a form required for comparison with the catalog of part data already resident in the robot data base. Part identification is then accomplished by seeking a match between incoming image data and stored part description data.

In many parts of the world, tremendous efforts are being applied to develop general vision systems; this includes projects at MIT, CMU, University of Massachusetts [34] and in Japan [89]. For example, Shapiro [87] has discussed the role of AI in computer vision, and has described some computer vision systems that can be called intelligent, such as the VISION [34] system designed by Hanson and Riseman and their students at the University of Massachusetts, the ACRONYM system [14] and the McGill Computer Vision System [52]. Shirai [89] has reported the Japanese "Advanced Robot Technology: JUPITER" project which aims at the research and development of an advanced robot that works in critical environments. In this project, visual sensing is the main theme of the sensing technology which is divided into four subthemes: sensors, feature extraction, scene description building, and model representation and matching. There was particular emphasis on the theories and technologies for low-level vision processing. Recently, many image processors have been developed for feature extraction. But, a complete general vision system generally consists of low-level, intermediate-level and high-level processing subsystems. Thus, accompanying the

research on low-level vision processing, there is also a need to develop high-level vision processing. This thesis is devoted to high-level vision processing.

2.2. Review of Work on Interpreting Line Drawings from a Single View

Research on three-dimensional scene analysis can be traced back to the early 1960's. Since that time, much attention has been devoted to interpreting line drawings automatically.

In the early seminal work of Roberts [77], the whole range of computer vision was covered. He first obtained a line drawing from a photograph, then matched it to a few 3-D models by transforming each model to it and finally interpreted the image. His matching process consisted of the selection of junctions corresponding to the vertices of the same object, and the matching of the junctions with the vertices predicted by the object model. Since his program was able to predict other views of the scene, it marked a significant break from pattern recognition by emphasizing descriptions of the objects present in a scene and the spatial relationships between them.

The next significant work was that of Guzman [33] on the segmentation of bodies in a scene containing polyhedra. His SEE program accepted a line drawing and produced output lists identifying and describing the bodies present in the scene by using a set of heuristic rules related to the types of vertices. The basic idea behind SEE is to make global use of information collected locally at each vertex: SEE combined different kinds of strong or weak evidence to make reliable global

judgements. The results were successful in decomposing even rather complex scenes of polyhedra. But his heuristics were very *ad hoc* and his program was intended only to partition the scene into bodies and provide this as input to a recognizer which might derive 3-d descriptions.

Huffman [40] and Clowes [22] stressed that the relationship between a scene and its image needs to be made explicit. Huffman classified lines that are the projections of edges into 3 types : convex, concave and occluding. Assuming that all images are taken from a general position, he showed that for a trihedral world, junctions could be catalogued into only 12 possible types. The consistent labelling of the lines in an image uniquely corresponds to a particular 3-D scene. If a picture has no possible labelling, it is impossible to realize it. Clowes determined a consistent interpretation by a search space technique.

Mackworth [58] described a program, Poly, that interprets line drawings of polyhedral scenes such as Guzman's SEE and Clowes' OBSCENE. The conceptual framework for POLY was inspired by Huffman's "dual-graph", which was presented as a device for checking an interpretation provided by the Huffman-Clowes labeling process.

Waltz [101] extended Huffman's method to line drawings with shadows. He added more types of lines for shadows and the illumination status of the two adjacent regions. Thus the number of interpretations of a line rose from 4 to 12, with a consequent massive number of possible junction labellings. Waltz enumerated physical interpretations of the junctions in line drawings of a trihedral

world and designed a filter program that usually converged to a single labeling in near linear time. It is remarkable that the filter program constrains the search space so successfully.

Turner [97] extended Waltz' principle to the analysis of scenes containing certain curved objects. He used two planes to approximate a curved surface in the vicinity of a corner. Thus a corner composed of both plane and curved surfaces might be approximated by a purely polyhedral one. Turner restricted curved surfaces to quadric ones and proposed a systematic way of constructing a junction dictionary including about 3000 junctions. Turner's labeling scheme is essentially an extension of the Huffman-Clowes scheme. Therefore, moving along a curved line, a possible labeling inconsistency may be encountered. Turner enumerated the cases of such label changes.

Another attempt in reconstructing curved bodies was made by Shapira and Freeman [82, 83]. For 3-D bodies with vertices formed by three faces, a cyclic-order property was defined. The property augmented the grammatical rules that govern the possibility or impossibility of the existence of 3-D bodies corresponding to 2-D line-structure projections.

Chakravarty [16] developed a generalized line and junction labeling scheme which deals with planar-faced or curved-surface solid bodies, which have vertices formed by three surfaces. In this scheme, 3 types of lines and 8 types of junctions were defined. By dealing with regions and lines, objects can be correctly labeled by the set of junction types. Illegal cycles in the junction transition graph were shown to correspond to impossible configurations.

As Barrow and Tenenbaum [9] pointed out, one caveat about using junctions for labeling line drawings of curved objects is that determination of the junction category may depend on subtle variations in geometry that are difficult to distinguish in practice. They pointed out that surface boundaries depicted in line drawings provide a good estimation of surface structure in the absence of other information. In a complete vision system information from contours must be combined with information from many other sources -- such as texture gradient, stereopsis, and shading -- to recover a more accurate and complete description of surface shape. They showed a three-step computational model for interpreting line drawings as 3-D surfaces, based on constraints on local surface orientation along extremal and discontinuity boundaries. The three steps are line sorting, 3-D boundary interpretation, and surface interpolation. Under the assumptions of surface smoothness, general position and curved torsion minimization, a specific space curve recovering technique was described.

Binford [13] has argued that procedures for obtaining image boundaries and interpreting line drawings are valuable, and that interpreting line drawings is not a dead-end approach which had been explored in previous years by Huffman, Waltz, and others, with assumptions that are too restrictive to be useful. Much of the previous work on the interpretation of image lines had concentrated on the constraints imposed on boundary junctions by certain classes of geometric objects. Lowe and Binford [54] argued that there are more general constraints on the formation of image boundaries, and that the use of these constraints is of great importance for the interpretation of real data and general classes of images. They

classified lines (intensity discontinuities) into edges, markings and shadows caused by geometric, reflectance, or illumination discontinuities. From general assumptions regarding occlusion by solid objects, the direction of illumination, aspects of object geometry and the production of illumination discontinuities by geometric discontinuities, some general constraints on the interpretation of lines were derived. They also developed a reasoning system for interpreting an aerial photograph of an aircraft which can operate efficiently with these incomplete constraints.

Although extensive work on interpreting line drawings from a single view has been done by many researchers, there still remain problems where further research is need. these include :

1. A more general labeling scheme. For line drawings, labeling is a useful tool for sorting lines and segmenting objects. Waltz' scheme which is based on Huffman-Clowes' scheme is restricted to the domain of polyhedra. Turner has extended the Huffman-Clowes scheme to curved bodies, and Chakravarty has further generalized a labeling scheme. However, Chakravarty's scheme is only for labeling a body not a scene, and its "C" type junction is artificial. The property of convexity of a line is useful for analysing a scene, but, it is ignored by Chakravarty's scheme. Both the Turner and the Chakravarty schemes are restricted to certain types of objects.
2. Combining the constraints imposed on junctions, lines and faces; combining the information from different sources; and combining the local cues with global cues. Barrow and Tenenbaum, Binford and Lowe have done some work on this field. However their work represents only a beginning .
3. Waltz' filter method is only one kind of model for human reasoning. For vision, a scene offers a lot of redundant information and a human seems more likely to use other reasoning methods. Simulating the

human vision activity, in an attempt to develop other effective programs is a worthwhile research topic.

4. All of the work mentioned above assumes a fixed vantage viewpoint. The automatic selection of vantage viewpoints is an important unsolved problem.
5. Based on segmentation and qualitative description of a scene, further quantitative description and construction of a 3-D model of the scene are interesting topics to be pursued.

In this thesis, based on Waltz' and Chakravarty's schemes, a modified and extended labeling scheme has been developed for scenes containing curved bodies and shadows. Instead of Waltz' filter method, an expert system which combines forward reasoning and propagation search has been designed and implemented to label images. A viewpoint planning scheme has also been developed in order to quantitatively describe a scene.

2.3. Review of Work on Analysing a Scene from Multiple Views

As mentioned in Chapter 1, an object must generally be observed from several directions in order to form an assessment of what it looks like or to form a 3-D model of it. In order to form a 3-D model of an object from sequential views, Underwood and Coates [99] developed a program which forms a 3-D description of a planar convex object when the object is rotated in space. In their match algorithm, the connections between surfaces, the number of edges which bound a surface and the clockwise ordering of edges form the deterministic factors. Preiss [70] described an approach which interprets a standard engineering drawing

of a planar object for construction of its 3-D representation. This approach consisted of three main steps: interpretation of projected faces, interpretation of dashed lines, and assembling them into a body. The connectedness properties and the geometric relationships between junctions or faces (such as coplanar relationships), were used for matching junctions. In this approach, the final 3-D representation of a body is complete, and consists of each of its faces, edges and vertices along with the three coordinates of each vertex. Later, Sakurai and Gossard [80] described an approach which creates the Boundary Representation-like solid model from three two-dimensional orthographic projections of an object. The views may contain straight lines and circular arcs, and the hidden lines and arcs must be identified. The solid model may contain planar, cylindrical, conical, spherical and toroidal surfaces.

Several researchers have investigated the problem of matching multiple views of a block world in front of a featureless background. Using wide-angle stereo, Ganapathy [29] designed a scheme which uses some heuristic rules, such as "Single Match", "Order Match", "Connectivity" and "Table Match", to choose an initial match between corresponding vertices in order to reduce the search space. His program finally stops after building up the 3-D coordinates of the vertices. The idea of using cues to accelerate the matching process has been extended in the prototype system which we have developed to form an expert system. Shapira and Freeman [82, 84] developed a program for constructing a description of solid bodies from a set of pictures taken from different vantage points. A heuristic procedure was devised for establishing matches between junctions in the

different pictures and determining the validity of doubtful junctions. It first establishes matches among junctions by using the constraints of projection and the connectedness between junctions; then it establishes matches among lines by using the cyclic-order property and fills in missing connections between junctions and missing junctions. The final description reported by the program involves bodies made up of face groups which are described in terms of triples of matched lines.

Some problems encountered in analysing multiple views are similar in many ways to the problems in motion analysis. In motion analysis, Badler [7] developed a system for describing 3-D motion from 2-D line drawing image sequence. Assuming that (1) the locations of point-like object features (such as vertices) are supplied with each frame, (2) the image sequences are relatively continuous, and (3) the objects in the environment are recognizable, the system translates sets of point displacements into a 3-D trajectory of the body. The significance of this system is that it uses spherical projection to model the physical appearance of an observed scene and it describes the body motion at the conceptual level, such as "moving left and away, rotating to the right", etc. Roach and Aggarwal [76] described a hierarchical matching system for images of polyhedra moving in 3-D space. The images were processed to extract line and vertex descriptions. These descriptions were then segmented into preliminary object interpretations based on general domain constraints. These interpretations were maintained by the system until conclusive evidence was obtained to decide which was correct. The correspondence between consecutive images was established by the hierarchical system which invoked the lower level processes only as the upper levels failed.

The top level process calculated a centroid for each object interpretation and using information from preceding images determined a predicted location for every centroid. These predictions were then matched by a nearest neighbor rule to the centroids found in the succeeding image. At the second level, coarse descriptions of relative object position, e.g., object A is to the left and below object B, were used to match object interpretations. The third and lowest level matching process was based on the relative positions of the polygonal faces in each interpretation. In this manner several different levels of processes used information from various object descriptions and relationships to establish the correspondence between images. Asada [4] developed a system which describes 3-D motions of jointed trihedral blocks. In this system, a Huffman-like labeling scheme and an object-to-object matching method are first used to segment the line-drawing images into individual blocks and to find the possible correspondence of their junctions between closely consecutive frames. A transition table of junction labels and contextual information is used to analyse structural changes of the line drawings. Then, the shape rigidity property of three vertices on a block is used to evaluate geometrical parameters, such as the 3-D coordinates of the vertices and motion parameters. The transition table of junction labels offers useful cues for matching junctions. The combination of a labeling scheme with a matching method reduces the search time. These two ideas are adopted in the prototype system which we have developed.

It is only in recent years that attempts have been made to match multiple views in a complex environment in order to incrementally construct some kind of model

of a scene. Herman, Kanade and Kuroe [37] described the 3-D MOSAIC project whose goal is to incrementally acquire a 3-D model of an urban scene from images. Their method is to first extract 3-D shape information from the images by stereo analysis, then to match two views based on junction matching and finally to generate an approximate model of the scene by using task-specific knowledge. Crowley [24] described a navigation system for an intelligent mobile robot which included techniques for the construction of a line segment description of a recent sensor scan and the integration of such descriptions to build up a model of the immediate environment in the form of a list of directed line segments. Herman [38] described an algorithm which matches vertices in two 3-D descriptions. The algorithm consists of three main steps: first, initial matches are obtained for each vertex based on local properties; next, a Waltz-filtering procedure is applied which propagates topological constraints to reduce the set of matches; finally, a tree-type search which uses both topological and geometrical constraints gives globally consistent sets of unique matches.

In comparison with single image analysis, the use of multiple views can be more practical and attractive than conventional two-camera stereo, when a series of progressive views is conveniently available, such as in the environment of a moving robot. But in order to implement a working system for a complex environment, more work still needs to be done in following areas:

1. The wide-angle stereo technique must be combined with a viewpoint selection scheme.
2. More powerful matching techniques are required. This includes: the selection of suitable features for matching, the combination of a

segmentation scheme with matching, and the finding of more constraint rules for matching.

3. The research domain must be extended to more complex bodies, scenes and environments.
4. Real scenes and imperfect line drawings must be dealt with.

In this thesis, multiple widely separated views of a complex scene need to be matched, and the 3-D models of bodies need to be constructed. Thus a hierarchical expert system for matching and model construction has been developed. It combines a labeling and segmentation scheme with object matching; and combines a variety of cues to form knowledge bases which not only accelerate the initial matching, but also the whole process of matching.

2.4. Representation of Solids and the World

2.4.1. Model of Solid Bodies

Methods for the representation of solids are important in computer vision, computer graphics, CAD and related areas.

Engineering drawings provide a representation scheme for solid objects which is traditional and widely used in spite of the fact that it is ambiguous. Usually drafting textbooks state that a **sufficient number of views, sections, details, and "notes"** should be supplied in a drawing to avoid "ambiguity". As Requicha [71] has pointed out, this "definition" is too informal for a precise study of drafting as a representation scheme.

Sometimes 2-D models are used to describe 3-D scenes. Usually 2-D models make the representation easy, but multiple models are necessary for analyzing a scene from different directions. Such 2-D models have been used for recognition of mechanical parts by Perkins [67] and for recognition of an indoor scene or an outdoor scene with constraints on the configurations of its objects, e.g. by Tenenbaum and Barrow [92].

Requicha [71, 72] has introduced a mathematical framework for characterizing certain important aspects of representations, such as validity, completeness and uniqueness. He has discussed seven basic methods which may be used to construct unambiguous representation schemes for bodies: these are pure primitive instancing, spatial occupancy enumeration, cell decomposition, Constructive Solid Geometry(CSG), sweeping, interpolation, and Boundary Representation(BR). In a spatial occupancy enumeration scheme, space is divided into a grid of 3-D cells (usually cubes), and a body is represented by a list of the cells which it occupies. The scheme is unambiguous, unique (except for positional nonuniqueness), and easy to validate, but it is quite verbose.

The CSG and BR methods are currently considered to provide the most important representation schemes. In boundary representations, a body is described by segmenting its boundary into a finite number of bounded subsets usually called "faces", and representing each face by its boundary edges and vertices. Faces must be represented unambiguously if a boundary representation scheme is to be unambiguous. Generally boundary representations are not unique and they are verbose. However, it is feasible to generate line drawings and graphic displays

from these representations. CSG representations are ordered binary trees. Nonterminal nodes represent operators, which may be either rigid motions or the operations of regularized union, intersection, or difference; terminal nodes are either primitive leaves or transformation leaves. CSG schemes are unambiguous but not unique. Any CSG tree is a valid representation of an r-set [72], if the primitive leaves are valid. CSG representations do not provide an efficient source of geometric data for producing line drawings of solid bodies.

In CSG representations, the information of relationships between subparts is implied in its operators. For object recognition, it is usually most useful to distinguish the various relationships between the subparts of a body and to describe them explicitly. To achieve this objective, two approaches have been developed. One approach is suggested by Binford [12]. His scheme is to model bodies in terms of "generalized cones". A generalized cone is defined by a 3-D space curve, called the "spine", and planar cross-sections of arbitrary shapes normal to this spine. Generalized cones are well suited to describing bodies which have a natural spine. But bodies produced by molding, beating, welding or sculpture tend to be awkwardly described by generalized cones. This representation scheme is not unique and there may be an infinite number of generalized cones representing a single body. This makes recognition of bodies difficult. The idea of using generalized cones to approximate parts of bodies after segmenting them into appropriate pieces has been further developed by Marr and Nishihara [59]. They proposed a hierarchy of models and discussed various index schemes, such as the specificity index, the adjunct index and the parent index, for characterizing the precision of the models and for accessing the components of a model.

From consideration of communication with ~~human beings~~ in natural terms, Agin [2] has developed a verbal modeling scheme to represent a body. His method is hierarchical, using generalized cylinders as primitive elements and their assemblies as higher level subparts and parts. There are three fundamental specifications for specifying the relationships among parts: snakes and stacks, attachment points, and position displacement. In this representation there are three classes of objects: prototypes, descriptions and instances. A prototype created by a programmer is used to represent a class of bodies. A description is a concise recipe for copying a prototype and assigning specific values to its variables to form an instance which represents a concrete body.

Another approach is the relational modeling technique given by Shapiro and Moriarty [85, 86]. This technique categorizes three-dimensional bodies at a gross level. The rough models have only three kinds of 3-D parts: sticks, plates and blobs. All three kinds of parts are "near convex." A relational description of a body consists of a set of parts of the body, the attributes of the parts, and a set of relations that describe how the parts fit together. Since data from two-dimensional views is generally rough and imperfect, Shapiro's idea of using rough models for recognition is interesting.

In many industrial environments man-made objects are assembled from parts or made using machining processes. Consequently, a solid representation which provides a good description of the structure of the solid should consist of volumetric primitives which contain the minimal structural information. If the primitives are not represented by rough models such as in generalized cones or in

relational models, a large amount of detailed geometric information needs to be included in the representation. Since, primitive parts are manufactured, measured, seen and graphically displayed by their surfaces, a face-based representation is more natural and suitable to describe them. This face-based representation should reflect varied shapes and make higher level structures concise and reasonable.

A surface representation scheme called the **Enhanced Spherical Image (ESI)**² was developed by Smith [90, 39] for body representation. An ESI model for a convex body consists of a set of vectors with unit length. The components of an individual vector represent the direction of a surface normal of the body. A scalar value, which represents the surface area corresponding to the normal vector, may then be tagged to each point on a unit sphere which corresponds to the vector. Using this model, rotations of a body about an arbitrary axis are analogous to spinning the sphere. ESI must satisfy the center of mass conditions, that is, the center of mass of the "weights" associated with the marks on the surface must be at the center of the sphere. Based on Minkowski's theorem [68] and the lemma developed by Pogorelove [69], the ESI representation for a convex body is unique up to a translation. It is feasible to use ESI as a basis for recognition [41] and determination of the attitude of a body [15]. Although the extension of ESI to the domain of smooth and non-convex bodies has been studied by Horn and other researchers [39], generally there still are difficulties associated with the representation of smooth and non-convex bodies.

²Sometimes this is called an Extended Gaussian Image (EGI).

For representing arbitrary surfaces, McPherson et al. [61] pointed out that curved surfaces can often be approximated by those surfaces described by low order polynomial equations. The total number of coefficients required to model an order K polynomial surface would be $(K+1)(K+2)(K+3)/6$. Given a general quadratic equation of a closed surface, the invariants which describe the shape, orientation and location of the surface can be mathematically derived.

There are many other solid representation schemes (e.g. skeleton models, oct-trees, etc.); we refer the reader to other reviews and original publications for more details [10, 36, 73, 71, 72, 12].

In the world around us, there are two kinds of man-made bodies. One kind is simple and has little or no structure (e.g., rectangular boxes and balls). The second kind is more complex and has a structure which is made up of simpler subparts (e.g., tables and chairs). Any efficient method for the recognition of classes of complex objects must make use of this structural information. Thus, an ideal body model needs to contain both:

1. Structural information, describing how the various subparts of the body are related, and
2. Geometric information describing the shape, volume and location of the body and its subparts.

Although the representation methods mentioned above have different characteristics and are used in various applications, most of them, such as traditional engineering drawings, oct-trees, cell decompositions and boundary representations, do not give explicit structural information. Others, such as skeleton models, generalized cones,

relational models and constructive solid geometry models, more or less describe the body structure, but the structural information usually must be given explicitly by the user.

The idea of describing the structural and geometric information of a body at different levels of its model and of automatically constructing the model from computer vision suggests the development of a new 3-D representation scheme: CSG-EESI representation, and of a corresponding conversion system.

2.4.2. Model of the World

Iyengar et al. [42] pointed out that we must increase the robot's capabilities to compensate for unpredictability in a world³. In this instance, because the spatial relationships between the robot and the world are no longer predetermined, two questions immediately arise. First, what does the world map look like? Second, where is the robot at any given moment relative to that world? This is called self-location. To answer these two questions, it is first necessary to model the world properly.

In Koutsou's paper [46], the six extant model-based languages were surveyed.

1. AL : Stanford University (**A**ssembly **L**anguage).
2. AUTOPASS : IBM (**A**utomated **P**arts **A**ssembly **S**ystem).

³In this thesis, we distinguish the words "environment" and "world". A world of a mobile robot consists not only of an environment in which the robot works, such as a workshop or a room, but also of all the bodies which locate in the environment.

3. RART : University of Edinburgh (**Robot APT**).
4. ROBEX : Technical University of Aachen (**Roboter Exapt**).
5. LAMA : MIT (**Language for Automatic Mechanical Assembly**).
6. LM : University of Grenoble.

In these systems, bodies are represented fully or partially. The world models are very different in these systems. In AL, it consists of information about position and orientation of bodies and it is generated and updated by a compiler. In AUTOPASS, the initial world model representing the initial state of the world and geometric information, is generated by the Geometric Design Processor, while a compiler updates it after compilation of each statement, so that the model represents the current world state. In PART, it is generated by "parsing" the source program and it contains information about the geometry of objects, their initial positions and orientations, the sequence of world states and the corresponding transitions. The world model is used by a compiler to infer the positions of objects throughout the assembly. In ROBEX, it represents the general environment in a particular manufacturing cell and not a particular operation to be performed. It remains unchanged for a whole number of tasks to be performed in the same environment.

Tsuji [95] described a mobile robot developed in Osaka University which monitors a building environment. The robot is initially parked at a standard position in the known environment and is then driven around via a given route. While moving around, it stops every few meters, takes pictures and analyzes the images to find obstacles and changes in the scene. The robot system uses a hierarchical world

model for guiding the navigation and image interpretation. At the top level of the world model, a 1-D route model represents the robot's route and the record of its past stops. At the second level, there is a 3-D local work space model which consists of the camera parameters and a map of the current block and the next few blocks to be viewed by the robot; this is used for image interpretation. At the lowest level, the object models specify the observation point for the robot and 2-D templates of the objects.

In order to facilitate the navigation and manipulation of a mobile robot, a model of its world should be constructed. A complete model of a world may contain information such as: (1) the model of an environment and the models of the bodies within it; (2) the layout of the world and the positions and orientations of bodies; (3) the relationships between bodies; (4) the available free spaces; and (5) the state changes of the world. A unified representation of the world model is desired, but until now it has not been developed. In our prototype system, the world model contains: (1) the model of an environment and the models of the bodies within it; (2) the planar layout of the world and the positions and orientations of bodies; (3) some relationships between bodies, such as "occluded by" and "touched by"; (4) the projection of the available free spaces on the floor; and (5) the trace of the path of the mobile robot.

2.5. Review of Work on Planning

Adaptive sensing and planning are essential for intelligent robots. An intelligent robot needs to plan its activities which include movement, sensing and thinking. As Ballard and Brown [8] pointed out, some skilled vision is actually like problem solving: vision for information gathering can be part of a planned sequence of actions; and planning can be a useful and efficient way to guide many visual computations. The visual processing certainly depends on planning. On the other hand, planning depends on the state of the world. A few of the automatic planners, which have been developed to date using AI techniques, are based on the assumption that the state of the world is known exactly at the time of planning. In an unknown world or a dynamic world, where this assumption is invalid, the state of the world should be explored by robot sensors.

2.5.1. Methodology of Planner

A planning process explores the states of the world which become known as a result of actions, and tries to find a sequence of actions that achieve predefined goals. Planning consists of three components: plan generation, plan decision making which includes plan selection and plan coordination, and execution monitoring. A plan must specify its expected results at each stage. If a departure is detected, the planner must be able to replan. Charniak and McDermott [18] pointed out that the design of a complete planner would have to address the following issues:

1. What is the correct notation for plans?

2. How are time maps produced and maintained?
3. How can problems (such as protection violations) be anticipated and corrected?
4. How do you manage the search through the space of possible plans?
5. How are plans translated into actual action in the world?
6. How do you monitor the progress of a plan?
7. How do you replan when things go wrong?

Graphically, the states of the world can be arranged in a tree with initial state as the root, and branches resulting from applying different actions in a particular state. Thus, in planning, intelligent search is essential. This search involves subgoal selection, action selection, and action argument selection. Intelligent search also implies a meta-level capability: the ability of a program to reason about its own plans.

Most plans have a hierarchical structure; each goal in a plan can be replaced by a more detailed subplan to achieve it. Although a finished plan is a linear or partial ordering of problem-solving operators, the goals achieved by the operators often have a hierarchical structure. The method of hierarchical planning has been implemented in a number of planning systems, including ABSTRIPS, NOAH and MOLGEN [8]. The hierarchical planners use a hierarchy of abstraction spaces to develop a plan. The method is to first sketch a plan that is complete but too vague and then to refine the vague parts of the plan into more detailed subplans until finally the plan has been refined to a complete sequence of detailed problem-

solving operators. The advantage of this approach is that the plan is first developed at a level at which the details are not computationally overwhelming.

If a problem solver knows how each problem-solving operator changes the state of the world and knows the preconditions for an operator to be executed, it can apply the means-ends analysis technique⁴ to solve problems. ABSTRIPS and most other planners use this technique. Means-ends analysis restricts the number of operators that apply to a goal, thus it reduces the amount of search space. However, there may still be several applicable operators for a goal and there is no way of knowing whether the subgoal of an operator can be satisfied or not. Reducing expensive backtracking is still an important issue.

NOAH (Nets of Action Hierarchies) was designed as part of the Computer-based Consultant project at SRI International, Inc. around 1975. NOAH plans by developing a hierarchy of subgoals which are called abstract operators. Abstract operators can not be executed until they are expanded to subgoals attainable by problem-solving operators. Operators are not ordered until a potential interaction is detected, and then they are ordered to avoid the interaction. Thus NOAH solves interaction problems constructively.

Sacerdoti [79] pointed out that experimental planning systems have been developed that display the following features:

⁴Briefly, the means-ends analysis technique involves looking for a difference between the current state and a desired state of the world and trying to find a problem-solving operator that will reduce the difference. This process continues recursively until the desired state of the world has been achieved.

1. plans can be generated at multiple levels of detail;
2. plans can be viewed as partially ordered sequences of actions with respect to time;
3. each action is expected to produce a single state change characterized by a single primary effect;
4. a plan is not generated at all unless the planner determines that it will be totally successful in meeting all specified goals;
5. simple plans requiring information gathering can be generated;
6. unsophisticated techniques for dynamic repair of unsuccessful plans during execution have been developed;
7. plans can be used to control robot devices with simple use of sensory feedback and simple replanning.

However, the integration of these existing capabilities into a single plan generation and execution system would in itself constitute a formidable and worthwhile research goal.

2.5.2. Trajectory Planning

In robotics, one important topic of planning is trajectory planning. Many researchers have done significant work in this area.

Thompson [93] described the JPL robot navigation system. In this system, the results of scene analysis are used to create and update a terrain model that is partitioned into map sectors of a convenient size. Within a map sector, terrain regions are described by polygonal boundaries which are represented as lists of the vertices of the polygon. The regions are classified as traversable, obstructed or unknown. The boundary between the loaded and not-loaded sectors is represented

in the map as a special obstacle. When the border is encountered by a planned path, the map will be expanded. A measure of path cost was used to define a selection criterion for optimal path search, and the A* algorithm that was developed by Hart and the Maze-solving method were applied to find the minimum cost path from a start position to the goal position.

Udup [98] described an approach to the trajectory planning problem for computer controlled manipulators with two movable links and multiple degrees of freedom. The manipulator is modelled as a sequence of connected cylinders, one each for the post, boom and forearm. Obstacles are approximated by polyhedra which constitute a hierarchy of abstraction spaces: primary map and secondary map. Planning begins with hypothesizing a trajectory. Following this are iterative steps that involve checking for collision and trajectory modification. Under normal circumstances, the loop terminates when a safe trajectory is found. Principles of hierarchy, separability and reversibility were used in the planning. For each goal, the strategy is to plan in the secondary map first and then to refine the trajectory in the primary map.

Considering that collision detection may be viewed as a sequence of intersection checks among appropriately defined static bodies, and at appropriate time intervals, Ahuja [3] developed two methods for detecting intersection among three dimensional bodies. The first method uses a conservative criterion to detect the occurrence of intersection. It was assumed that bodies may be represented by polyhedra. Each polyhedron is uniquely described by the coordinates of its vertices, and the adjacency relationships among them. The projection of a

polyhedron on any plane is determined by the projections of the vertices, using the original adjacency relationships. The criterion is that if a plane, which is any one in a given set of planes, is found in which projections of two bodies do not overlap, then noninterference between the two bodies is guaranteed. For this method, when the projection of a body is nonconvex, a description in terms of convex polygons must be extracted or the nonconvex polygons must be decomposed into a set of convex polygons. The second method uses oct-trees to represent three-dimensional bodies. In the oct-trees a label of a node is black or white if it corresponds to a block which is completely contained within the body or the free space respectively. The criterion to determine the occurrence of intersection is that two bodies intersect if there exists at least one pair of corresponding nodes in the two trees such that one of them is black, and the other is black or gray.

Giralt et. al. [32] described a multi-level planning and navigation system for the mobile robot HILARE. The system design can be viewed as decisions through multiple cooperating expert modules together with a high-level coordinator in a hierarchical means-ends structure. The expert modules have their expertise in a variety of overlapping domains, e.g. object identification, navigation, exploration, itinerary planning. The modules consist of: specialized and redundant knowledge bases, algorithms and heuristics, local error-processing capabilities, and communication procedures. Modules may access one another as primitive action-units. The coordinator activates modules based on a means-ends analysis of the current situation. The world model is contained in the production system data base. The model is a hierarchy of body-centered concepts of the form

(name feature pattern-body)

which describe space and bodies. Space is defined by places (e.g., rooms, work-areas), frontiers (e.g., doors), and locations. Places have the property of surface convexity and are connected to other places by means of frontiers. Locations are elements within places which can be identified by point coordinates, relational descriptions, or feature descriptions. Bodies are defined by features (e.g., shape, colour and dimension) based on sensory perception. The obstacles were defined as polyhedra, whose floor projections fully determined the navigation problem, and which could be located either by initial information or by robot perception. The planner applies a cost function to find an optimal or ϵ -optimal path.

Lozano-Perez and Wesley [55] described a collision avoidance algorithm for planning a safe path for a polyhedral body moving among known polyhedral obstacles. The algorithm is essentially an extension of the visibility graph algorithm which finds collision-free paths for a moving point by finding the shortest path in a visibility graph [64]. The mechanism added in the Lozano-Perez's algorithm is growing the obstacles and shrinking the moving body to a point. In the algorithm, it was assumed that all bodies are modeled as sets of, possibly overlapping, convex polyhedra. The extended visibility graph algorithm no longer guarantees optimum paths among three-dimensional obstacles. It has been used to plan collision-free trajectories for a manipulator with seven degrees of freedom; these trajectories have been successfully executed in the laboratory.

Later Lozano-Perez [57] extended the above algorithm to the configuration space approach. This approach is based on characterizing the position and orientation of

a body as a single point in a configuration space in which each coordinate represents a degree of freedom in the position or orientation of the body. The configurations forbidden to this body, due to the presence of other bodies, can then be characterized as regions in the configuration space, called configuration space obstacles. The Visibility Graph Algorithm can be directly extended to deal with three-dimensional configuration space obstacles. But the approach has some important drawbacks. Shortest paths in configuration space that move along the boundaries of the configuration space obstacles are very susceptible to model inaccuracy and position error. This problem can be alleviated by adding a uniform "safety margin" around the obstacles, but doing so might disqualify some feasible paths. An alternative heuristic (but complete) path searching strategy was also developed by Lozano-Perez [56]. This heuristic algorithm is suboptimal, and uses the obstacles' complement (i.e. free space) for path search. The free space is represented by a hybrid cell tree which is based on a generalization of the quad-tree representation. The path finding algorithm first searches the hybrid cell tree and produces a list of empty Configuration space cells that touch or overlap, and enclose the start and goal configurations, then chooses a piecewise linear path contained in the cells.

Rueb and Wong [78] described a method of structuring the free space of a roving robot's environment into a set of overlapping convex regions which is represented by an attributed hypergraph. The convex regions are chosen so that every boundary is a constraint imposed by an obstacle wall. Finding a shortest path between two stops is then equivalent to searching for the shortest path in a

weighted graph. Since the number of wall segments in the worst case is the square of the number of obstacle walls, the time complexity of the construction method is at least $O(N^2)$, and the time complexity of finding the shortest route between two points may go up to $O(N^4)$ time by using Dijkstra's algorithm.

Weiss et al. [102] formalized an analytical approach to dynamic robot visual servo control systems by casting position-based and image-based strategies into classical feedback control structures.

Recently Iyengar et al. [42] proposed a heuristic method that enables a mobile robot to navigate in an unexplored terrain. The information of the terrain is learned from multiple journeys on which the start and destination points are given. The model of the terrain consists of a 2-D spatial graph, which specifies the experimental routes and stops, and a Voronoi diagram. This algorithm allows the optimal continuous transition from local to global path. But, the complexity of the algorithm and the method of exploration were not given in this paper.

Different models of the world, different models of robots, and different evaluation functions affect the selection of trajectory planning methods. For simplification of trajectory planning, the model of the world usually needs to be simplified as an oct-tree or a 2-D map, and the robot needs to be simplified to a simple figure, or even to a point. In this thesis, the trajectory planning is subject to the viewpoint planning. The model of the world and the robot are simplified to a 2-D map and a point. For improvement of the planned trajectory, the above mentioned methods or other methods, such as Lozano-Perez' algorithm, may be used.

2.5.3. Viewpoint Planning

Autonomous control of mobile robots requires that they can decide by themselves what to see, where and when to go, and what to do and how to do it. Thus view planning is a primary issue. As Tsuji [95] has mentioned, most vision systems for autonomous vehicles do not continuously view the environment but analyze images taken at each stop. There are a few exceptions which are equipped with real-time vision systems to accept consecutive images sent from TV cameras, however their capabilities are limited to the performance of fixed tasks, such as finding obstacles within a specified range [94]. Thus, the viewpoint planning is the main component of planning views.

In history, the "stationing watchmen" problem has received certain attention. For one of its restricted domains -- "Art Galleries", Chvatal [21] has proved that for every polygon with n vertices there exists a decomposition into at most $\lfloor N/3 \rfloor$ disjoint star-shaped polygons. In the case of orthogonal art galleries, Kahn, Klawe and Kleitman [43] showed that $\lfloor N/4 \rfloor$ watchmen are sufficient. But, for general situations, it is a NP-hard problem.

The "2-D visibility from a point" problem has been addressed in several papers. Freeman and Loutrel [28] described an algorithm for the solution. This algorithm can determine which parts of a known simple polygon are visible from a vantage point which is either outside or inside the polygon. But its time complexity is not clear, and it is at least nonlinear. Asano [5] devised an $O(N \log N)$ algorithm to solve the problem for a set of N line segments. Suri and O'Rourke [91] also

devised an $O(N \log N)$ algorithm to compute visibility polygons in the presence of a set of line segments where two line segments do not intersect except perhaps at their endpoints; the $O(N \log N)$ algorithms have been proved as optimal by reduction from the problem of sorting n positive integers. Two linear algorithms have been devised by ElGindy and Avis [26], and by Lee [51] for solving the visibility problem from a point inside a simple polygon.

A star-shaped polygon P has the special property that within it there exists a point z such that from z all points p of P can be viewed. Thus, an important aspect of planning viewpoints is related to the decomposition of free spaces into star-shaped polygons. Although partition of a simple polygon with holes into the minimum number of star-shaped polygons is an NP-complete problem [44], Avis & Toussaint [6] developed an $O(N \log N)$ algorithm for partitioning a simple polygon into at most $\lfloor N/3 \rfloor$ star-shaped polygons. This algorithm consists of 3 steps:

1. Obtain a triangulation T of the simple polygon.
2. Colour the vertices of T with colours (1,2,3).
3. For $i=1,2,3$, output each vertex with colour i together with a list of all adjacent vertices. These vertices form a decomposed star-shaped polygon.

Thus, this algorithm always yields a decomposition with at most $\lfloor N/3 \rfloor$ star-shaped polygons. However, it does not normally give a decomposition into the minimum number of star-shaped polygons. Keil [44] developed an $O(n^5 N^2 \log n)$ algorithm to find the minimum star-shaped partition of a simple polygon, where n and N are denoted as the number of the vertices and the number of notches in the simple polygon.

Although the important problem of planning the next "best" viewpoint has not received much attention so far, there has been some related work. Kim et. al. [45] described an approach to determine camera positions for successive views while looking for the distinguishing features of objects; they made the following assumptions:

1. objects are stationary;
2. the camera can be placed at any desired position and direction;
3. object models are known *a priori* and are stored in a database;
4. an object recognition process which is capable of identifying, locating and determining the orientation (when possible) of objects is available.

In this approach, the distance and direction of the camera are determined separately. The distance is determined by the size of the object and the features, while the direction is determined by the shape of the feature and the presence of occluding objects.

Connolly [23] described two algorithms which use partial oct-tree models as input to determine the best next view when there are no prohibited sensor positions and the sensor points toward the fixed origin for all views. The first algorithm sets up a sphere around the scene. The sphere is sampled along latitude and longitude and the sample point on the sphere which covers the largest unseen area is selected as the next best viewpoint. The second algorithm counts the area of the faces, which are common to both *Unseen* and *Empty* nodes in each of six directions corresponding to coordinate axes, and selects the three maxima to form a direction vector which determines the next view.

Turchan and Wong [96] describe a method for environment model acquisition. In this method, features extracted from range data are encoded as an attributed graph. The strategy that they used to place the next position of the robot is to directly locate it in the front of a "Pseudo-Boundary".

Robots have restrictions on their movements. Thus the location of vision sensors cannot meet the conditions required by Kim or Connolly. The simple strategy used by Turchan and Wong requires many more viewpoints than necessary. In Chapter 5 and 6, this issue has been investigated and corresponding algorithms have been developed.

2.6. Expert Systems

Expert systems defined by Nau [62] are problem-solving computer programs that can reach a level of performance comparable to that of a human expert in some specialized problem domain. The distinguishing feature of most expert systems is that they have a separate knowledge base which is manipulated with a separate control strategy. The knowledge base consists of the domain-specific problem-solving knowledge which is usually procedural. This kind of knowledge could be represented as a conventional computer program. However, in the situation where the precise series of steps necessary to solve the problem is not known, the knowledge often adopts the form of operators or pattern-invoked programs. One type of pattern-invoked program of particular interest is the production rule. Procedural knowledge can also be represented in first-order predicate logic. The programming language PROLOG is an example of such an approach.

Newell and Simon [63] pointed out that the production system is a good candidate for modeling human cognitive processes. The most important reasons are as follows:

1. Production systems have the computational generality of universal Turing machines.
2. Production systems can be incremental, since new rules can be created and added.
3. Under certain assumptions, the data-base models the functional characteristics of human short-term memory, and the production rules provide a possible model for human long-term memory.

In a simple production system, the productions are arranged in order with highest priority first. The control mechanism sequentially fetches a datum from the data base and a rule from the knowledge base, and then tests whether they are matched. If they are matched, the rule is applied. After any rule is applied, testing begins again from rule 1. Of course, control strategies may not be so simple. More sophisticated strategies could be adopted such as (1) state-space search which includes data-driven search, goal-driven search and graph-searching (e.g., breadth-first search, least-cost-first search, or heuristic search), (2) relaxation, and (3) problem-reduction. In production systems, rules may be partitioned into sets or hierarchies, and rule selection may use some special filters or apply meta-rules to choose one particular rule set.

Georgeff [30] pointed out that knowledge about the effects of sequences of actions, methods and plans is an important component of many problem domains. In placing constraints on production invocation, control reduces the amount of

interaction between knowledge units. In his paper, a general production system architecture, called a controlled production system was described. In the formal model of this architecture, control knowledge is abstractly represented by a language over the production set. However, in implementing the model, an implicit rather than explicit representation of the control language will usually be used. The fundamental feature of the formalism is that it separates the procedural control from the search strategy, which determines the order of selection of productions. If instructions are viewed as productions with a true condition, and predicates as productions with a null action, the standard functional and procedural languages are formally a restricted class of controlled production system. In the case where the control language is regular, a controlled production system is simply a representation of a flowchart for a sequential algorithm.

Chandrasekaran et. al. [17] proposed a conceptual hierarchical structure for medical diagnosis. In this structure, the successors of a concept node stand for subconcepts which refine that concept. Each concept node is associated with a set of procedures (experts) which attempt to apply the relevant knowledge to decide on the applicability of the concept to the case at hand. Part of such decision-making in a node is often the decision to turn the control over to subconcepts and their associated procedures to check on their applicability. The tree structure limits the communication between experts. An expert can only communicate via the super and subconcepts. This structure fetches a body of knowledge at an appropriate level of depth; therefore such an approach would be most suitable for solving problems in knowledge-rich domains.

Gilmore and Puraski [31] summarized the characteristics and capabilities of eight of the most popular expert system tools in nine commercial available expert systems: ART, Duck, Knowledge Craft, KEE, KES II, MI, Rule Master, and SI. For a better review of early and more recently developed expert systems, we refer the reader to the book by Harmon and King [35].

Expert systems provide an important tool which is being widely used in a variety of areas. In computer vision, which is a knowledge-rich area, the hierarchical structure combined with expert system tools offer a good foundation for the construction of a comprehensive vision system.

Chapter 3

Incremental Construction of 3-D Models from a Sequence of Framed Views

3.1. Introduction

In this chapter, we describe a system which incrementally constructs 3-D object models of an office or warehouse scene from planned multiple views. In particular, we address the matching and construction of 3-D partial models. The system which we have developed for incrementally constructing 3-D models of objects is illustrated schematically in Figure 3-1.

The incremental construction of object models from planned multiple views involves the following principal elements:

1. decomposition of a framed view and construction of partial 3-D descriptions of the view;
2. matching of partial 3-D descriptions of a view with the built-in model of the robot environment;
3. matching of partial body descriptions derived from the current framed view with those partial models constructed from previous views;
4. identification of the new information in the current view and the updating of the models;
5. identification of the unknown parts of the models which are being constructed so that further viewpoints can be planned;

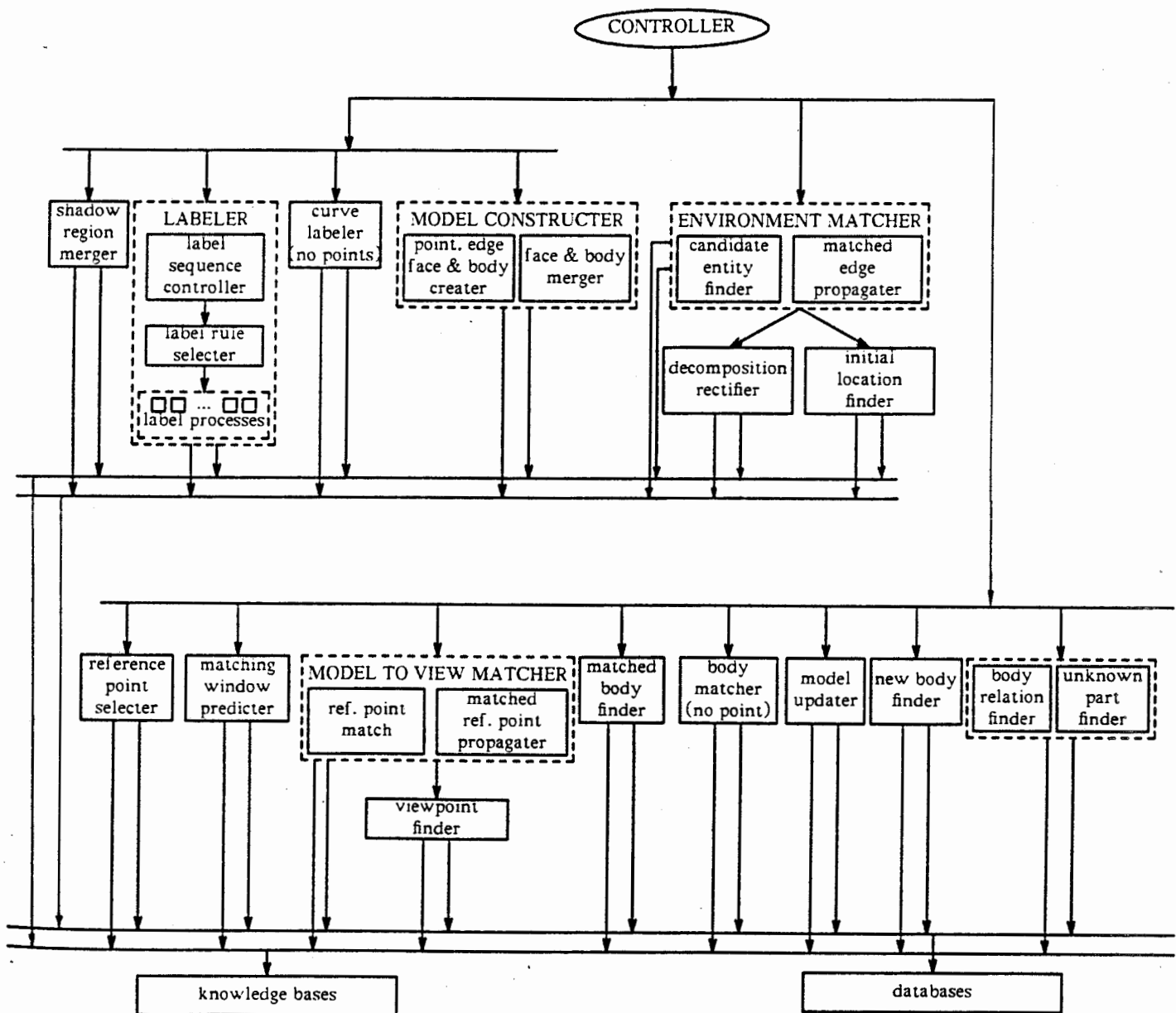


Figure 3-1: A schematic sketch of the system for matching and model construction.

6. determination of the relationships between object bodies and the environment and construction of a partial map of the scene.

In the following sections we discuss elements 1 - 5 in turn. Each of these elements is more or less related to matching.

In this approach, matching is based on the rules (constraints) which have been derived from geometry, topology, photometrics, physics, triangulation and problem assumptions, as well as being derived from the uniqueness and consistency of features. These rules are set out in detail in Appendix B and form the knowledge bases of an expert system. The rules accelerate the matching process and guarantee reliable matchings. Thus our goal here is not to try to exhaust all rules for matching, but rather to establish and test the methodologies. Features lacking sufficient evidences might not be matched with only two images, but selected multiple views would ensure their identification.

Strategies for recognition can be data-directed (bottom-up), knowledge-directed (top-down) or some mixture of the two. In our approach, a bottom-up and top-down mixed strategy is used to match partial 3-D descriptions of a view with a built-in model of the robot environment and a data-directed strategy is used to incrementally construct 3-D body models for the objects in the environment. We are interested in exploring how far we can go with a data-directed strategy.

In the scene-learning process, the robot vision system generally produces incomplete and erroneous knowledge of the objects in the environment; consequently, it is important to identify the unknown parts, to correct the erroneous knowledge and to assimilate new information.

3.2. Labeling Scheme

Starting from the Chakravarty [16] and Waltz [101] labeling schemes, a modified and extended labeling scheme has been devised for labeling scenes containing shadows and certain curved objects. The scheme is based on classifying junction types that may occur in a view of a scene. In the scheme, lines are classified as occluding lines (labeled as "1"), internal convex lines (labeled as "2a"), internal concave lines (labeled as "2b"), concave boundary lines (labeled as "1b"), limb lines (labeled as "1m"), or shadow lines (labeled as "0").

Junctions are classified into 12 types: v, w, y, p, t, k, m, x, a, s, q, and c types, which are shown in Figure 3-2. The label of a junction is of form $\alpha\beta S\gamma$. It consists of four parts: the first numerical part α indicates the number of regions of a body part at the junction; the second numerical part β indicates the number of region of another body part at the junction, when two body parts form the junction; the third symbolic part S indicates the type of the junction; the fourth numerical part γ indicates the sub-class of the junction. The junction labels and the corresponding line labels are tabulated in Appendix C. An example of a labeled scene is shown in Figure 3-3. Junctions, that do not belong to the 12 types, are "peculiar"; they occur in situations such as special alignments or accidental alignments, and they are not labeled.

As in Chakravarty's scheme, this scheme attaches a label to each line of a junction giving information about the associated faces, and is able to verify impossible configurations based on the local properties of junctions.

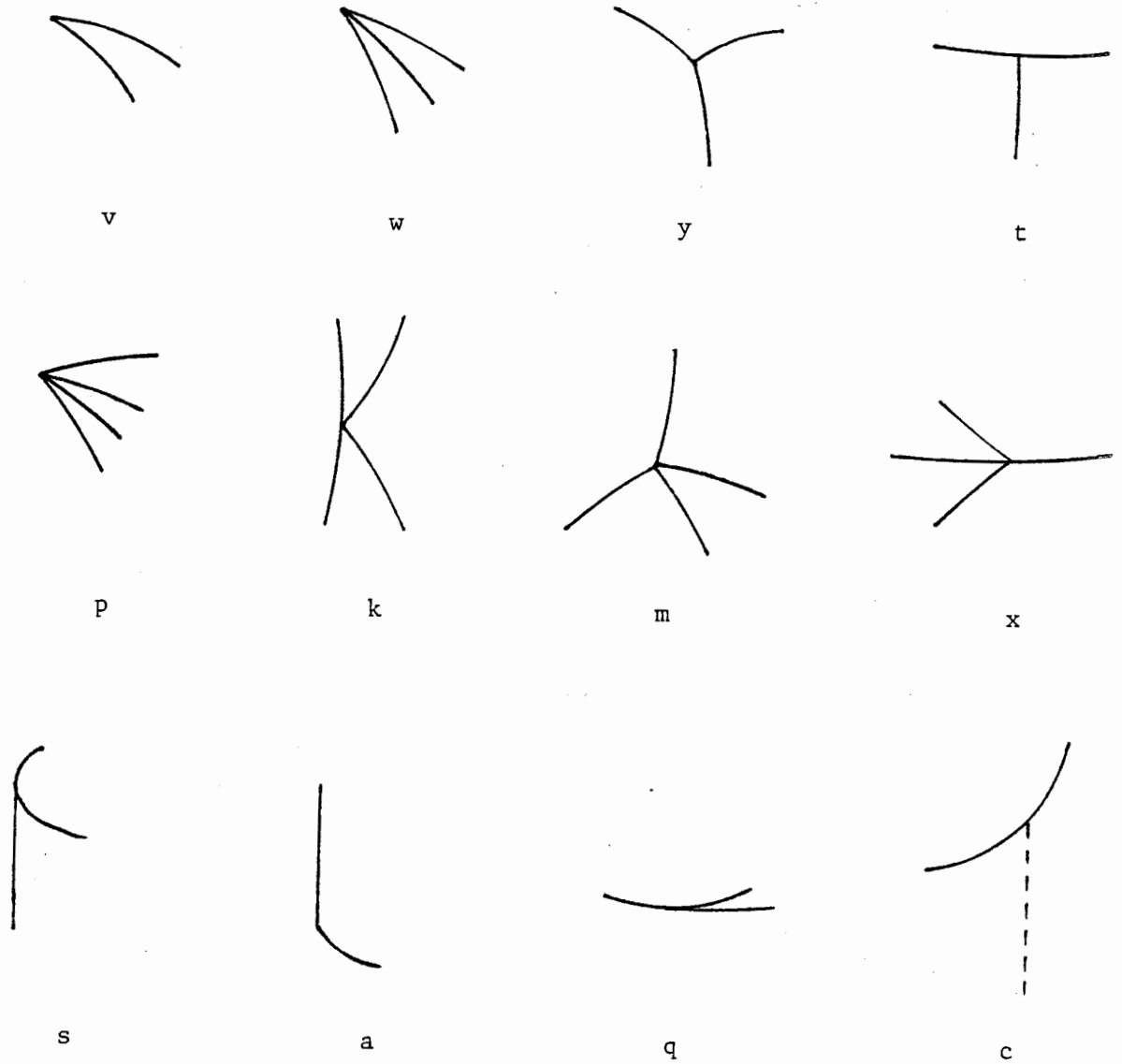


Figure 3-2: The types of junctions.

3.3. Decomposition

For each view, the partial 3-D descriptions of bodies are derived by labeling and segmenting the image. In general, the decomposition process first merges the regions that are separated by shadow lines⁵, and then it labels the junctions and lines in the image. After labeling, internal representations are created for real vertices, edges and faces. At that time, those edges separated by virtual junctions are combined, and the partially visible edges and faces are identified. In the last step, faces are combined on internal edges to form bodies. Some relationships between bodies, such as "touched by" or "occluded by" will also be identified⁶. Thus, hierarchical internal representations which are used as partial 3-D descriptions are constructed for each body in a view.

The labeling scheme described in the last section is used here for labeling scenes containing shadows and certain curved objects. The images are labeled by a sub-expert system; knowledge of labels (production rules) is stored separately in micro-knowledge-bases according to the categories of related junctions. The knowledge and rules are set out in detail in Appendix D. The top level of the sub-expert system controls the sequence of the labels. It first arranges the addresses of the junctions that need to be labeled in an "ORDER QUEUE". The junctions which are generally easier to label, such as "p", "w" and "y" types, are arranged at the

⁵Under the assumption of point light source mentioned in Chapter 1, shadow lines are generally detected.

⁶Occluding edges indicate the "occluded by" relationships, and concave boundary edges imply the "touched by" relationships.

front of the "ORDER QUEUE". When a junction and its related lines have been successfully labeled, the junction is deleted from the "ORDER QUEUE". Meanwhile, its immediately adjacent junctions will be inserted in a "PRIOR QUEUE". The top-level of the sub-expert system then propagates the labeling from the labeled junctions to their immediate neighbours, thus the label procedure can take advantage of derived facts and the labeling time can be reduced. The second level of the sub-expert system selects the appropriate micro-knowledge-base, according to the category of the junction given by the top-level of the sub-expert system, and sequentially selects the production rules from the selected micro-knowledge-base in order to label the junction and its related lines. At the lowest level of the system are the processes which carry out the following tasks:

1. fetching the related facts from the appropriate micro-databases;
2. matching the facts with the condition of a rule;
3. incorporation of the resulting labels into the appropriate micro-databases; and
4. adding the addresses of the immediate neighbours of the successfully labeled junction into the "PRIOR QUEUE".

The decomposition is conservative, i.e., it favors the separation of objects on concave edges. If a curve has no feature point, it is given a label according to its convexity: a concave curve is designated as a concave boundary edge and a convex curve is designated as a convex internal edge. The errors caused by an incorrect decomposition are expected to be corrected by facts collected later or by the knowledge stored at higher levels.

3.4. Matching the Environment Model

The initial location of the robot in the environment is not known *a priori*. In order to determine the initial coordinates of the robot in a fixed coordinate system keyed to the environment, it is necessary to identify which entities in a framed view correspond to parts of the environment model. For this purpose, at least some edges of an entity should be matched with a connected part of the environment. Edges are stable relative features which contain dimensional information and they are at the lowest level (except for vertices). Since the geometry (shape and dimensions) of the environment are known, an edge-based matching process has been devised for matching the environment.

The process first matches the completely visible real edges of entities from a framed view with the built-in environment model; this is done according to their attributes, the categories (e.g. planar or conical) and the directions of their adjacent faces. The edge attributes concerned consist of category (e. g., straight line, circle or other curve), type (e.g., shadow, occluding boundary, concave internal, convex internal, concave boundary, clipping line or limb), convexity, and approximate length. If an edge in a view is matched with several edges in the environment model, then a "matching confidence" will be assigned to it which is proportional to the inverse of the number of matched pairs. An entity is considered to be a candidate for part of the environment model if at least its visible and well labeled internal and occluding edges match with edges in the environment model. From these candidates, entities will be designated as being parts of the environment on the basis of the following properties:

1. at least two matched edges,

2. a maximum number of matched edges, and
3. a maximal sum of confidences for matched edges.

Following this identification, a top-down analysis process, which propagates the matched facts according to the built-in model of the environment, will be applied to those entities in order to:

1. Further verify the matched facts and find more matching facts. If in the propagation an inconsistent fact is discovered, then the initially matched entity will be rejected.
2. Identify the matched vertices and determine the position of the current viewpoint of the robot.
3. Correct the results of the decomposition of the current view. When those concave edges which were initially labeled as the concave boundaries are revealed as the concave internal edges of the environment, their labels are revised and the corresponding bodies are merged together.

Since the 3-D coordinates of two known feature points and their spherical coordinates in a view can be used to determine the position of a viewpoint, the position can be determined from any two matched edges. From a pair of matched edges and the approximate depths of their related points (e.g. end points), the process identifies the best two pairs of matching points. Since the 3-D coordinates of the environment points are known, the possible position of the current viewpoint can be calculated from the two pairs of points. After another pair of matching edges has been discovered by the matching propagation procedure or when the other pair of matched edges is used for propagation, the new facts

will be used to confirm or correct the position of the viewpoint and make it more precise.

3.5. Matching Partial Body Models

Once the environment model has been matched, the partial 3-D descriptions from the first view will be used as the initially constructed partial models of the bodies in the scene.

In order to match the partial descriptions of bodies derived from a new view with those partial models constructed in previous views, a multi-level feature matching approach has been used. This approach first matches the partially constructed models to the 3-D descriptions in the current view by selecting those reference vertices from the object models and the environment model which have the following features:

1. they are valid vertices or Shadow Intersection Points (SIP)⁷;
2. they are within the new view frame (though sometimes they may be occluded and unseen);
3. for each reference vertex, either the directions of the two constituent faces are known or the projection of the vertex is a boundary point of an unknown area in the current constructed map;
4. they are related to the objects of current interest.

⁷A Shadow Intersection Point (SIP) is an intersection point between two shadow lines which corresponds to a 0v junction or an intersection point where a shadow line crosses an edge (e.g., form a "2x1" type junction).

Following the selection of the reference vertices a prediction process determines the possible matching windows in the new view; this is done on the basis of the approximate position of the new viewpoint and the coordinates of the reference vertices which may only have approximate values stored in the models. The process finds the candidates for matching in the new view which are located within the matching windows. The window sizes are determined by the tolerance errors of the robot movement, the errors of the coordinates of the reference points and the relative distances of the new viewpoint and the reference points.

In the knowledge base, there is a "Junction Family Dictionary" which is listed in Appendix E. In the dictionary, each family consists of the possible junction types for a specific kind of vertex when it is viewed from different positions. Using the Junction Family Dictionary, and the categories and directions of the constituent faces, the matching process assigns each candidate a confidence. The candidate which has the uniquely highest confidence will be chosen as the matched vertex for a reference vertex, and its corresponding faces will be considered as matched faces. After finding a matched pair of vertices, the matching process propagates the fact along the emanating edges to adjacent vertices. A depth-first search is used at each pair of matched vertices, and when partial edges, unmatched vertices caused by occlusion or already matched facts appear, the match propagation for that direction will stop. Thus, different levels of features (i.e. faces, edges and vertices) can be matched in the same propagation process.

For those constructed body models which do not have any vertex or feature point (e.g. SIP), the edges and faces will be chosen as the basic matching

elements. According to their categories, types, shape parameters and approximate positions, the corresponding feature elements can be found. Also the matched facts will be propagated to their related feature elements.

3.6. Model Updating

After the faces have been matched, matched bodies can be derived from these faces. Following this, the model updating process searches the matched facts starting with high level features and moving to low level features; the low level features which do not exist in the partial body model are now filled in from the known parts of the current 3-D body description which has been matched. After matching the partially constructed models to those 3-D descriptions in a new view, unmatched bodies in the new view are identified. In order to test whether these are newly discovered bodies, a reverse direction matching process is used to check whether any vertex in an unmatched body has a corresponding vertex in the body models or the environment model. If this is not the case, then the body is new and is added to the database of the body models; otherwise the appropriate matched model will be found. Meanwhile, features separated in the new view or in the constructed models may be merged into one if their correspondence is unique in one of the two 3-D representations. The related revision will also be done.

In practice, data gathered by a robot vision system always includes certain tolerance errors. Although the relative positions of the viewpoints can be derived from a robot servo system, this information is generally imprecise. Since any two

views form a pair of wide angle stereo images, the matching process provides the information necessary to calculate the position of the sensor (the robot camera) quite precisely. This information can be used to correct the position calculated by the movement control servos and used for dynamically adjusting the robot movement.

3.7. Identification of Unknown Parts

The ambiguities caused by special alignments and accidental alignments can generally be distinguished by using multiple views. For example, when a strange junction type occurs in a view, if from other views the matched points belong to the same family of junctions, then it is caused by a special alignment, otherwise it is caused by accidental alignment. The ambiguities caused by accidental alignments can be ignored. For a special alignment, the ambiguity can often be solved by a correct decomposition, though sometimes higher knowledge of the scene may be needed.

Inside a body, self-occlusion may result in unknown occluded parts. Between bodies, an occlusion may cause the occluded bodies to be unidentified. For these two cases, unknown parts only occur at the occluding edges. Besides, a concave boundary edge indicates that the two related bodies are touching, and hence the touching parts cannot be seen if there is no means to change the status of the bodies.

In the system described here, when a model of a body has been created, only the internal, occluding and concave boundary edges which are the real edges of the

body are created. The model also contains a list of its boundary edges and a list of the bodies which occlude it. When a newly discovered surface is added into a body model, it is necessary to change the types of those occluding edges in the body model, which are matched with the edges of the added surface. These edges become the internal edges of the body and are deleted from the boundary list of the partial model of the body. Thus boundary occluding edges of a body model always indicate the self-occlusion of parts and the need for further attention.

The "t" type junctions caused by occlusion are kept in the input image databases. Although they are not the vertices of a body, they are important points for the construction of a map of the scene and for discovering the unknown parts caused by occlusion. For an occluded body, the search for its occluded parts is accompanied by a search for its "t" type points and those incompletely seen edges and surfaces which relate to the "t" type points.

All of the above outcomes will be organized and analyzed by a view planning system in order to further resolve the ambiguities. This component has not yet been implemented.

3.8. Experiment

Except for two modules: "Body Matcher (no point)" and "Unknown Part Finder", the system for matching and constructing 3-D body models shown in Figure 3-1 has been implemented using C-PROLOG under the UNIX operating system on a VAX 11/750. PROLOG is a programming language based on Kowalski's procedural interpretation of Horn clause predicate logic. It was originally developed

at the University of Marseilles as a practical tool for logic programming. C-PROLOG is a PROLOG interpreter written in the language C. From a user's point of view the major attraction of PROLOG is the ease with which clear, readable, concise programs can be produced; PROLOG is particularly useful in expert system applications. However, the current version of C-PROLOG and its interpreter has shortcomings, the main one being that its execution is rather slow. This is due to its lack of a good file system. Sequential access of data from databases slows down the whole computation.

Figure 3-4 shows two synthesized views from the scene shown in Figure 1-1; they have been successfully analyzed by the system described above. The movements of the robot, the depths of points and the plane normals gathered from the images generally have errors which have been assumed to have a Normal distribution. In the test, the images were first correctly labeled, except for concave intersection lines between the walls and the floor; these were not recognized as internal concave edges and thus their corresponding junctions were not labeled. At that stage, no evidence from the images could indicate whether these intersection lines were internal or boundary edges. Based on the labeling, the images were decomposed into bodies. Also, because of lack of evidence, the legs of the table were separately decomposed with the top of the table. The 3-D descriptions of bodies in the two views were then created. After that, the partial door of the first view had been matched with the corresponding door in the environment models, and thus the environment had been identified and the robot's location had been determined. The uncertainties and mistakes of labeling were

revised by the knowledge of the environment model, thus walls, the ceiling and the floor were correctly recognized. The partial models of bodies of the first view matched with the corresponding 3-D descriptions of the second view, and then the information of the new surfaces was added into the body models. Two legs and a box newly viewed from the second view were discovered and added into the database of body models.

3.9. Chapter Summary

Under the assumptions described in the introduction, the system described in this chapter can incrementally construct 3-D body models in an office or warehouse environment by matching planned multiple views. No prior knowledge of the objects is required by this system. The system includes the following important features:

1. a framed view is decomposed and partial 3-D descriptions of the view are constructed;
2. partial 3-D descriptions of a view are matched with the built-in model of the robot's environment;
3. partial descriptions of bodies derived from the current framed view are matched with those partial models constructed from the previous views;
4. the new information in the current view is identified and the models are updated;
5. the unknown parts of the models which are being constructed are identified so that further vantage viewpoints can be planned.

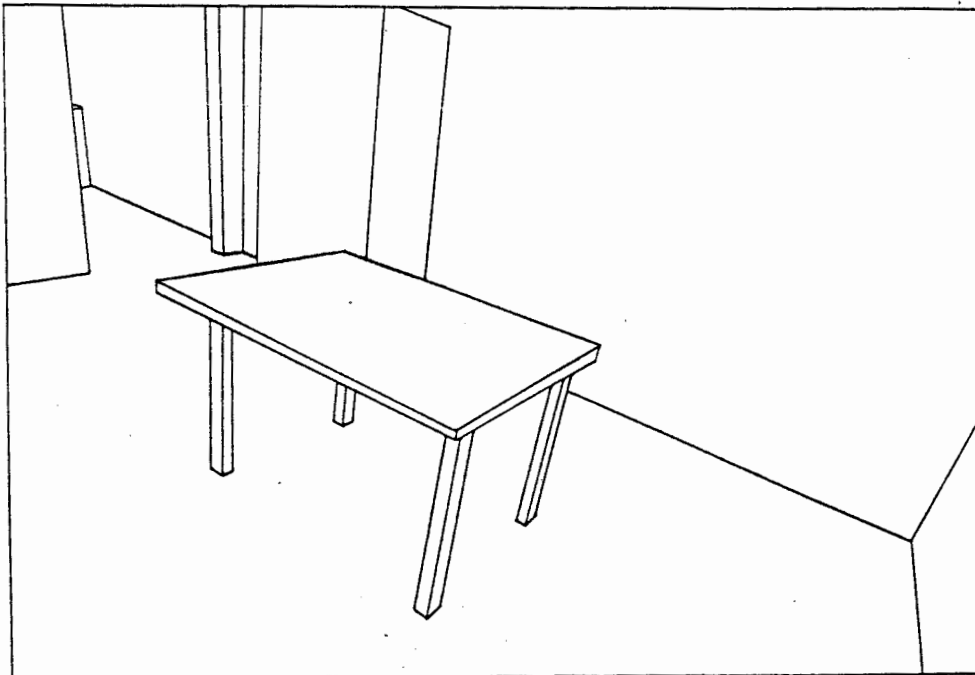
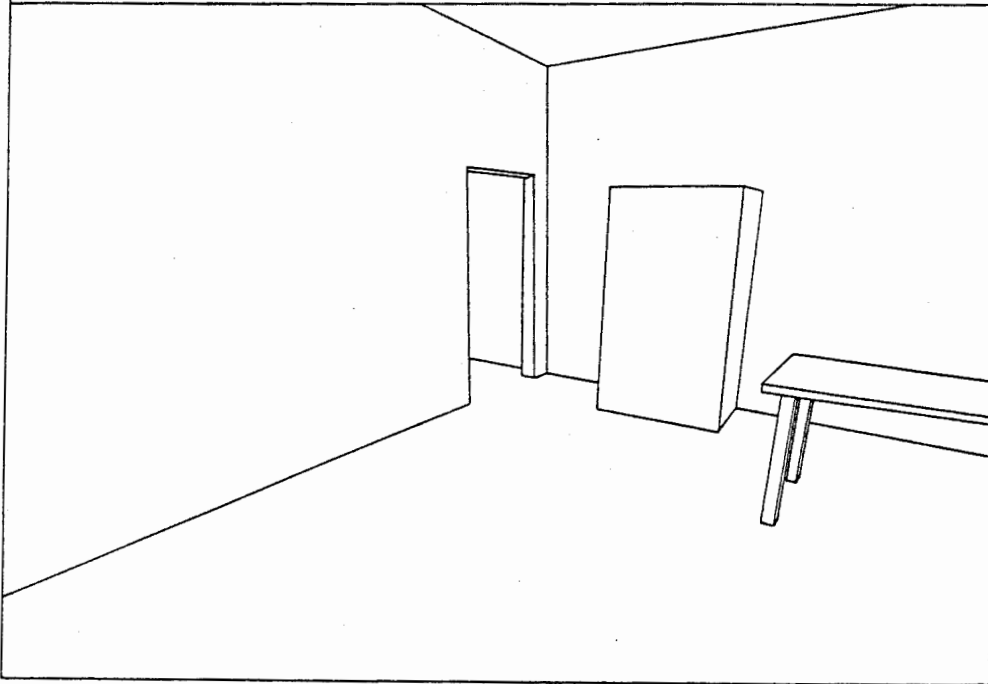


Figure 3-4: The two synthesized views which have been successfully analyzed by the system for matching and model construction.

As noted above, the system has been implemented in C-PROLOG under the UNIX operating system on a VAX 11/750, and has been tested successfully with synthesized images.

Chapter 4

Construction of the Environment Map

4.1. Introduction

In Chapter 1 it was stated that the shape and dimensions of the robot's environment are given *a priori*. However, the existence of unknown bodies affects the appearance of the environment and can change it with time. Thus, the mobile robot needs to explore its environment to develop the knowledge necessary for navigation and for manipulation of bodies.

The positions of bodies, the spatial relationships between bodies and the outline of the free spaces can be roughly described by a 2-D map. In Chapters 5 and 6, the map constructed by previous views will be used to plan the new viewpoints and the navigation routes of the robot. At the same time, the information gathered from new views is used to update the existed 2-D map.

In Chapter 3, it has been shown that the positions of the robot can be determined from perceived views and partial models of bodies can be constructed. In this chapter, we will describe an approach which creates an approximate projection map from the constructed body models (either partial or complete) and the given environment model.

When the environment model is given, the projections of the walls on the floor form the regions which limit the activity of the robot; and the frontiers indicate the pass-ways between the regions. In Chapter 3 Section 4, we mentioned the environment matching and the determination of the robot position. Thus, here we assume that these two problems have already been solved. Since the walls usually consist of planes, their projections are easy to calculate. Otherwise the creation of their projections can be handled as described in Section 2.

In Section 2, the method which calculates or approximates a silhouette for a known body represented by the CSG-EESI representation (refer Chapter 7) is described. This method consists of two parts: (1) the creation of the silhouette of the difference (or union) body from the silhouettes of its two parts, (2) the creation of the silhouette of a simple body which has an EESI representation. A method which successively constructs a 2-D map for a scene consisting of partially known polyhedron bodies is sketched in Section 3. This method uses the algorithm developed by Suri and O'Rourke [91] to find the visibility polygon from the outlines of the projections of partially known bodies. Finally, an approximation method which successively constructs a 2-D map for a scene consisting of partially known curved bodies is described in Section 4. This method is based on two ideas: (1) successive approximation and (2) the general assumption of continuity. It first approximates the silhouette of a quadric face by the silhouette of the corresponding complete quadric surfaces; the approximated silhouette of the quadric face is only updated when a cross-section on the quadric face has been perceived. By using a display system, the outlines of object projections, the projection of the

environment model, the viewpoints and the routes of the robot can be drawn on a screen or a hard copy device.

4.2. Construction of a map for a scene consisting of known bodies

Here we ignore the effects of holes which pass through a body on the map construction. Accordingly, we define the peripheral silhouette of a body as the outermost closed curve of the silhouette of the body.

In Chapter 8, we will introduce the CSG-EESI representation and its conversion expert system which converts BR-like representations into the CSG-EESI representation. Thus for a complete body model, we assume its CSG-EESI representation is available⁸. In the CSG-EESI representation, a complex body is formed by its simple parts. In this section, it will be obvious that the projection of a complex body can be formed by the projections of its simple parts.

First we consider how the union and difference operations affect projections:

1. If a body is the union of two parts, then its silhouette is the union of the silhouettes of the two parts.
2. If a body is the generalized difference [72] between a convex primary part⁹ and a secondary part, then, depending on the relative positions of

⁸Here the torus is simply not considered, since it is not convex. It could first be approximated by planes, and then be dealt with.

⁹If the primary part is not convex, then situations b and c are not always true. A counter-example is shown in Figure 4-3.

the two parts, the peripheral silhouette of the body is designed as follows:

- a. It is the peripheral silhouette of the primary part, when the silhouette of the secondary part is totally inside the silhouette of the primary part.
- b. It is the peripheral silhouette of the intersection boundary between the primary and secondary parts, when the silhouette of the secondary part totally overlaps the silhouette of the primary part. A simple example is in Figure 4-1.
- c. When part of the peripheral silhouette of the primary part is overlapped by the silhouette of a secondary part, this part of peripheral silhouette should be replaced by a part of the peripheral silhouette of an intersection boundary between the primary and secondary parts which should be closest to the replaced one. A simple example is in Figure 4-2.

Then we consider how the projection of a simple part which has an EESI representation can be calculated:

1. For a known plane-faced convex body, we can design an algorithm that first determines the "visibility from top" for each face of the body based on its normal direction and then finds the set of edges, each of which is adjacent to a "visible" face and an "invisible" face. These edges form the silhouette of the body.
2. For a convex body which is formed by a convex quadric surface and a number of planes which are in the extension of the quadric surface, an algorithm for calculating its peripheral silhouette is designed as follows:
 - a. Create the peripheral silhouette of the quadric surface as the initial approximate peripheral silhouette.
 - b. Arrange the planes in order.
 - c. Take the first plane P_1 from the order and consider a convex

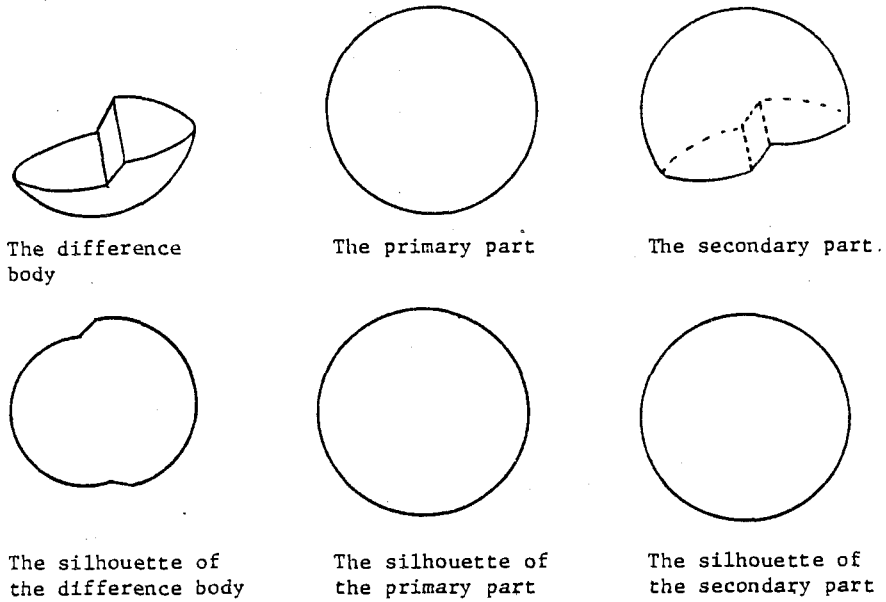


Figure 4-1: The construction of the silhouette of a difference body when the silhouette of the secondary part totally overlaps the silhouette of the primary part.

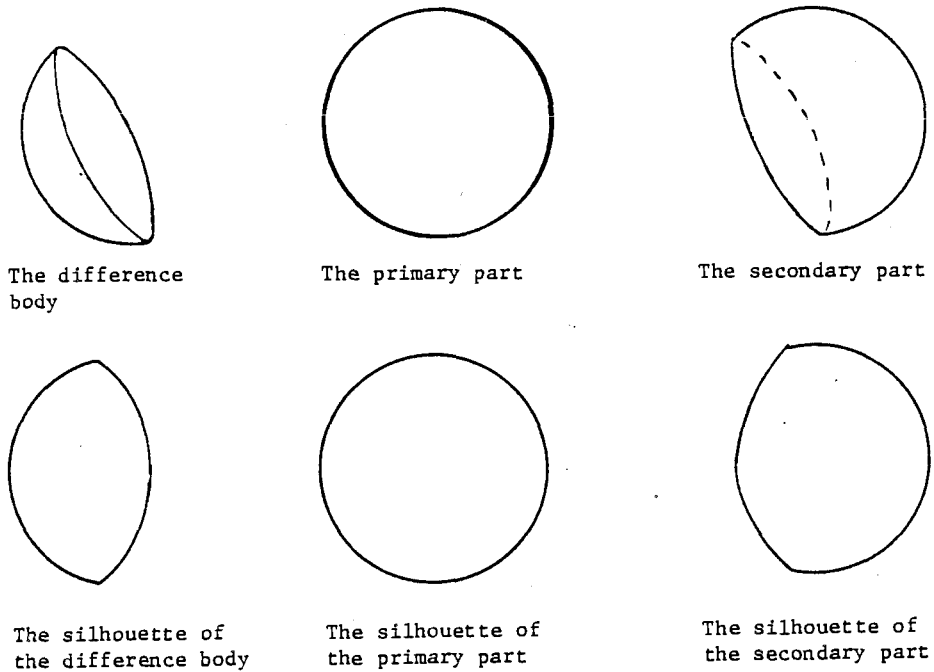


Figure 4-2: The construction of the silhouette of a difference body when a part of peripheral silhouette of the primary part is overlapped by the silhouette of the second part.

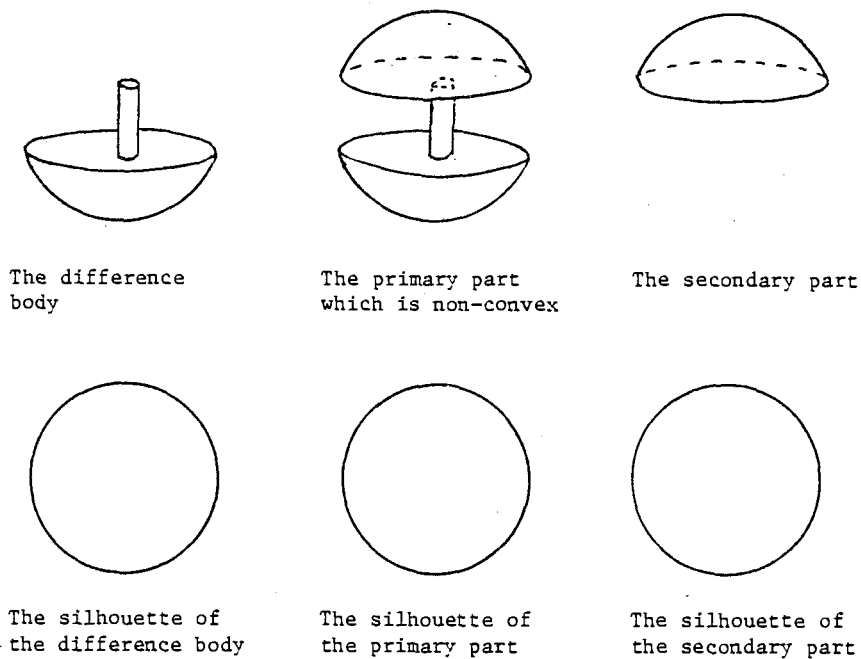


Figure 4-3:

A counter-example for the difference operation affecting projections when the primary part is not convex.

body D_1 which is formed by the quadric surface and the plane P_1 . According to the direction of the plane's normal and the relative position between the plane, the symmetric center (or axis) of the quadric surface and the projection plane, the silhouette of the convex body D_1 can be easily determined.

- d. Take the next plane P_n from the order and consider a convex body D_n which is formed by the body D_{n-1} and the plane P_n , where $2 \leq n \leq N$. N is the number of the planes. As a special situation, it can be considered as the difference between the body D_{n-1} and a body which is formed by the quadric surface and a plane that is co-planar with the plane P_n and has the reversed normal. The peripheral silhouette of the difference between these two parts can be calculated.
- e. Repeat the above step for each plane in the plane order until none remain.

3. For a body with some externally visible planar holes and/or concave cuts which are made by removing convex planar shapes from the previous two kinds of bodies, we can use¹⁰ the difference method described above to create the peripheral silhouette.

Thus for all bodies which have EESI model representations (except for torus surfaces), we can calculate the peripheral silhouette. In CSG-EESI representations, if a body has concave quadric surfaces, then the regularized difference operators are applied between the related subparts; otherwise, the union operators are applied to combine the subparts together. By using the union and difference operations described above, we can calculate or approximate the peripheral silhouettes of the bodies which have CSG-EESI model representations. Therefore, if the objects and the environment model are known *a priori*, according to their CSG-EESI models the corresponding 2-D map can be constructed.

4.3. Construction of a map for a scene consisting of partially known polyhedron bodies

If a scene consists only of polyhedron bodies which are partially known, then the 2-D map can be constructed by the following method:

1. For the first view, vertices and "t" type points of all bodies are projected onto the X-Y plane, then the projected points are connected according to their corresponding edges, and the outlines of projected edges for each body projection are figured out. The matched environment model is also projected onto the X-Y plane. After that,

¹⁰Here the planar holes and/or concave cuts are separated on the body that can be viewed as an union of several convex sub-bodies, on each of which there is only one hole or cut. Thus, the difference method still applies.

from the projection of the viewpoint, construct its visibility polygon [5, 91]. In the meantime, the confirmed and the pseudo boundary edges are distinguished. A confirmed boundary edge must be a visible edge corresponding to a viewpoint; the pseudo boundary edges are Projective Viewlines (PVL) which are added to connect the confirmed boundary edges to form the visibility polygon. At last, by using the display system, draw the outlines of object projections, the projection of the environment model (after the first view has matched the environment model), the viewpoint and the PVLs in order to form the initial partial map (See Figure 4-4).

2. For the successive views, the newly discovered vertices and "t" type points which are identified in the matching process described in Chapter 3 are projected onto the X-Y plane; the projected points and the existing points are connected according to their corresponding edges. Based on the old outlines and the newly added projected edges, the new outlines for each body projection can be figured out. After that, from the projection of the viewpoint, construct its visibility polygon. Which old PVLs should be deleted and which new PVLs should be added is determined according to methods described in Chapter 6 Section 3. By using the display system, the existing map can be updated by modifying the body outlines and PVLs, adding in the new viewpoint and the corresponding robot route which are selected by the method described in Chapter 6 (See Figures 4-5 to 4-9).

4.4. Construction of a map for a scene consisting of partially known curved bodies

It has been assumed that our research domain includes planar, cylindrical, conical and spherical surfaces. If a scene includes bodies that have quadric faces, it is less easy to get the projection map, since to construct the map, it is necessary to determine the projections of those curved bodies. In a Boundary-like representation, a body is represented by its enclosing faces, which are represented

in terms of such primitive entities as unbounded mathematical surfaces,¹¹ their boundary curves and vertices. For a curved face, the direction of a boundary also needs to be indicated in order to distinguish the two different parts of the corresponding quadric surface.¹² The mathematical description of a quadric surface or a curve and the construction of this description from vision is discussed in Appendices F and G.

A curve can be approximated by piece-wise line segments. The goodness of the approximation depends on the selection of points on the curve. The boundary of a curved face could be described by a set of such feature points as vertices, end points and extreme points. In general, an extreme point of a curve is sensitive to the orientation of the viewline with respect to the fixed coordinate system. But, if the robot stays at the same place and only stretches up its neck, the extreme points will keep the same ϕ values for the curved line in the two views. Therefore their 3-D coordinates can be easily calculated by triangulation. In some situations, certain calculating points¹³ could be imaged on a boundary to help the determination of its parameters or its approximation.

¹¹Here we distinguish the two concepts of surface and face: "face" is referred to as a bounded part of a surface; and "surface" is unbounded or complete.

¹²We define the direction of a boundary to be given by a set of unit vectors, each of which passes through a point on the boundary, and is co-linear with the tangent of the boundary at that point. According to the right hand rule, the face always lies on the side pointed to by the thumb, when the other fingers are along the direction of the vectors.

¹³Definition: The calculating points are those imaging points on a body which are chosen as feature points to determine the geometrical properties of the body (e.g., an extreme point may be used as a calculating point.)

Although the equations of a quadric surface and its boundaries can be calculated by suitably selecting enough points on them, to get the exact projection of a body with quadric faces onto a plane is still a tough geometric problem, especially when the body models are only partially constructed. Therefore, it is essential to design a reasonable approximation method to handle curved faces and their boundary curves or to handle their projections.

In this section, we describe an approximation method to construct the projection of a curved body which is partially known. This method is based on two ideas. One involves a successive approximation. Since quadric surfaces are well defined symmetric surfaces and the projection of a complete quadric surface identifies the projection of a cross-section which passes through a symmetric axis or center of the surface, it is reasonable to first approximate a quadric curved face by a cross-section which pass through the symmetric axes or center of the body. A quadric face can be considered as the remaining part cut from the corresponding surface. Its projection also can be approximated by cutting some parts off from the projection of its corresponding surface. The other idea involves the assumption of continuity; i.e., if there is no other contradictory evidence (e.g., a concave edge which indicates that the face intersects the other face), a face or an edge will keep its category and extend to an unknown area. This method approximates the projection of a curved body with the projection of a convex curved body that consists of planar and quadric surfaces which may be cut by planes.

The detailed method for creating a projection of a curved body which is partially known is described as follows:

1. For a straight line edge on the body, project its vertices and the "t" type points into the X-Y plane and connect the projected points by a straight line segment.
2. For a quadric face of the body:
 - a. Create its boundary projection. For a straight line boundary, the creation is as the above Step 1; for a curved boundary, first select some feature points on them, then project the feature points and end points on the X-Y plane, and connect the projection points in the original order.
 - b. For a spherical face, create a polygon(e.g., an octagon) which approximates the projection of a sphere in the X-Y plane. If there are circular boundaries on the face which indicate cross sections on the corresponding spherical surface, then for each cross section, use the difference operation described in Section 2, to successively approximate the projection for the remaining part cut by this cross section.
 - c. For a cylindrical or conical face, project a cross-section of its corresponding cylindrical or conical surface that passes through the axis of the surface and the vector, which is the vector production of the axis and the Z axis of the fixed coordinate system, on the X-Y plane, to form a first approximation of the face projection. If there are circular or elliptical boundaries of the face indicating cross sections on the corresponding surface, then for each cross section, use the difference operation described in Section 2, to successively approximate the projection for the remaining parts cut by this cross section. Further, if on the cylindrical or conical face there exist two convex straight line boundaries which indicate a section cut on the corresponding surface, then for each of such section, using the difference operation described in Section 2, to successively approximate the projection for the remaining part cut by this section.

After the projections of partial bodies have been created, as in the case where a scene consists of polyhedron bodies, it is necessary to get the outlines, to

construct the visibility polygon and to form the initial partial map for the first view.

For successive views, where the scene does not consist only of polyhedron bodies, we need to keep the constructed projections for those quadric faces and to check whether there is new evidence which indicates new cuts on a quadric face. If this is the case, then the corresponding difference operation should be used to calculate further approximate projections of the quadric face. Otherwise, the approximate projection of the quadric face will remain the same. After that, new outlines should be re-calculated; the visibility polygon related to the new viewpoint needs to be constructed; and the existing map should be updated accordingly.

4.5. An Example

For the office scene shown in Figure 1-1 and the viewpoints of the robot shown in Figure 7-2, the partial maps of the world which are shown in Figures 4-4 to 4-9 can be sequentially constructed according to the methods described. Although, the scene consists only of polyhedron bodies, the example offers an intuitive view of the methods and shows that they are feasible.

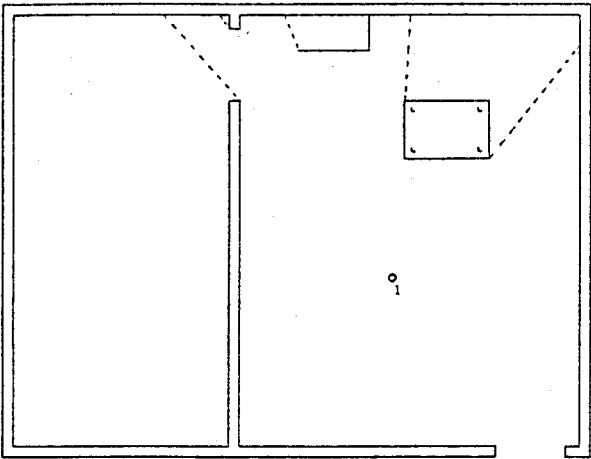


Figure 4-4: The partial map of an example scene constructed from Viewpoint 1.

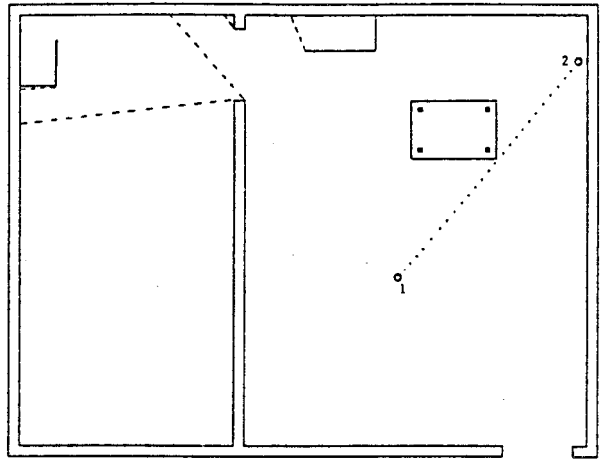


Figure 4-5: The partial map of an example scene constructed after Viewpoint 2.

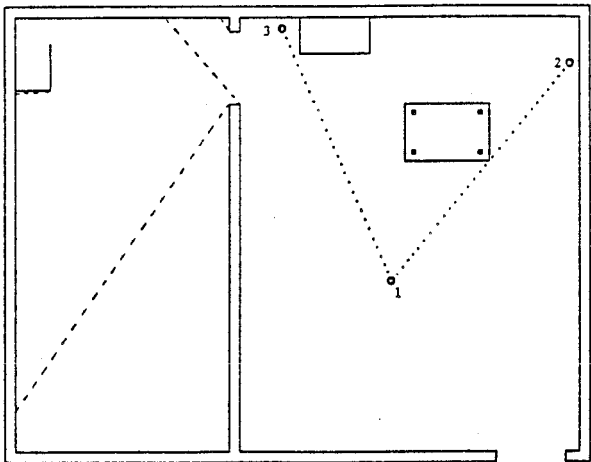


Figure 4-6: The partial map of an example scene constructed after Viewpoint 3.

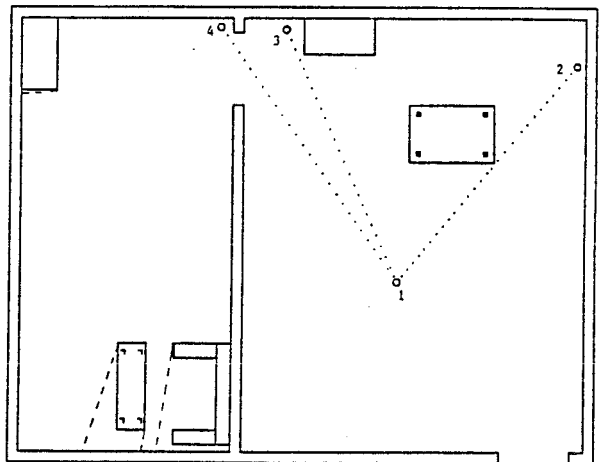


Figure 4-7: The partial map of an example scene constructed after Viewpoint 4.

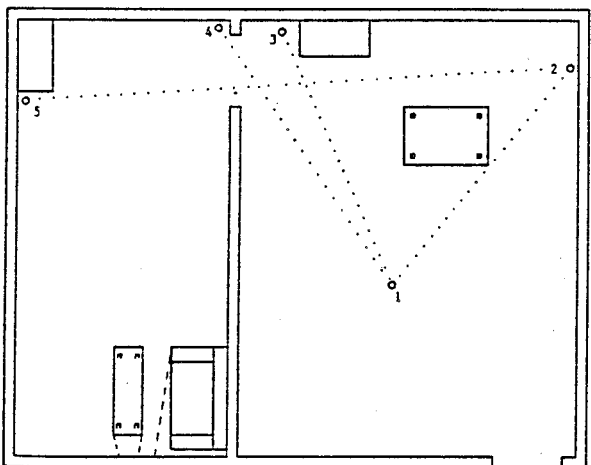


Figure 4-8: The partial map of an example scene constructed after Viewpoint 5.

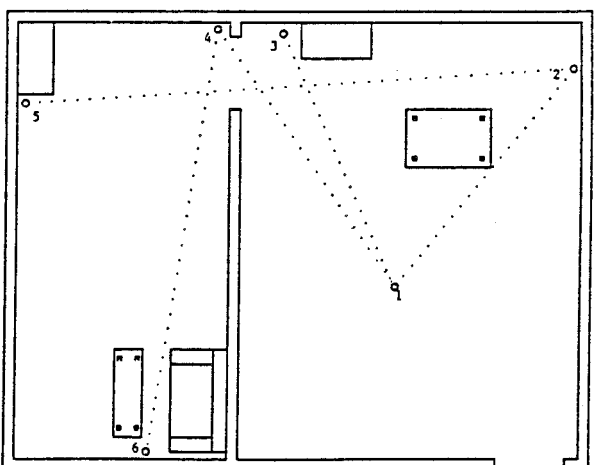


Figure 4-9: The partial map of an example scene constructed after Viewpoint 6.

----- PVLs.
 The trace of the robot path.

Chapter 5

Planning Viewpoints and the Navigation Route of a Patrol Robot in a Known 2-D Environment

5.1. Introduction

A patrol robot often works in a situation where the shape and dimensions of the surrounding environment (e.g. a warehouse) and the bodies within the environment are known. The task is to use feasible movement paths to observe and check whether or not everything is normal. Thus it is important to select optimal viewpoints and arrange corresponding navigation routes which are as short as possible. This problem is developed as a preamble to the more general problem where the bodies within the environment are unknown. On the other side, after the robot is acquainted with its environment, this problem will be appeared.

The factors which affect the selection of viewpoints include: the geometric properties of the bodies and their surrounding environment, the movement capabilities of the robot, and the geometric and perceptive properties of the sensor. To limit the scope of the immediate research problem, we have made the following assumptions:

1. The problem has been simplified to a 2-D case.

2. The objects in the environment are static, rigid, and weakly externally visible polygons.
3. The robot is only allowed to move around objects in order to view them (i.e., it cannot move over or under an object).
4. The sensor used to acquire the information has a negligible size.

In terms of computational geometry, planning the best viewpoints in the above 2-D case is equivalent to finding the minimum number of star-shaped polygons to cover a simple polygonal region with holes which are defined by the 2-D objects and their environment. In general, this problem is NP-hard, as has been proved by O'Rourke and Supowit [66]. Aggarwal [1] has shown that even if the polygon does not contain holes, the problem of finding a minimum covering with star-shaped components remains NP-hard. Also Keil [44] has shown that the problem of partitioning a simple polygon with holes into a minimum number of star-shaped polygons is NP-complete. Thus there is a need for a collection of fast near-optimal algorithms for solving the above problems.

This chapter first describes two heuristic approaches to planning viewpoints in 2-dimensions. The first $O(N \log N)$ approach is based on the static partition of a weakly simple polygon¹⁴ into star-shaped polygons. The second $O(N^2 \log N)$ approach is characterized by the sequential selection of viewpoints; this results in covering the edges of a weakly simple polygon with star-shaped polygons and may require fewer viewpoints than the first approach. Once the viewpoints have been

¹⁴It likes a simple polygon, except its edges may coincide.

selected, their order implies a corresponding navigation route, but the route is usually not optimal. For a patrol robot, it is necessary to reorder the sequence of the viewpoints and arrange the navigation route accordingly. By decomposing the free spaces into simpler components and connecting viewpoints with pass-ways, the problem of arrangement of the navigation route can be reduced to the well known NP-hard Traveling Salesman problem. Thus, it can be solved by one of the known approximation algorithms, such as Christofide's Heuristic algorithm in $O(N^3)$ time.

5.2. Planning Viewpoints in 2-D

Since the problem of the partition of a simple polygon into star-shaped polygons can be solved in polynomial time, the idea of the first approach is to form a simple polygon which can represent the free spaces. If the simple polygons, which represent the object models and the environment, are ordered and each polygon is connected with its successor by linking one of its vertices to a vertex of its successor, then it will form a single weakly simple polygon which represents the free spaces.

According to the Chvatal's theorem [21], we can claim that:

1. In the 2-D case, if all object models and their positions in the environment are known, there exists a plan which needs at most $\lfloor (\sum N_i)/3 \rfloor$ viewpoints to check everything, where $\sum N_i$ is the sum of the vertices of all polygons which represent the objects and the environment.
2. In the 2-D case where the environment can be ignored and the object models and their positions are known, a big triangle may be used to enclose all objects, and there exists a plan which needs at most $\lfloor (\sum M_i)/3 \rfloor + 1$ viewpoints to check the objects thoroughly, where $\sum M_i$ is the sum of the vertices of all polygons which represent the objects.

In the simplest situation, a known 2-D convex object could be inspected thoroughly from two viewpoints if they were far enough from the object.

5.2.1. Partition Scheme

From the above discussion, it can be seen that an approach to planning viewpoints with assumed knowledge of both the objects and the environment could consist of the following steps:

1. Order and connect the simple polygons to form a unique weakly simple polygon in $O(N \log N)$ time, where N is the total number of vertices of the polygon so formed.
2. Decompose the polygon so formed into star-shaped polygons by using the Avis and Toussaint algorithm [6] in $O(N \log N)$ time.
3. Find the kernel of each star-shaped polygon by using the algorithm offered by Lee and Preparata [48] - time is $O(N)$.
4. In each kernel, determine a viewpoint.
5. Remove those inserted edges which only decompose triangles from the formed weakly simple polygon if the third vertex of the triangle can be viewed from an adjacent viewpoint. Also remove the viewpoints in those triangles. This can be done in $O(N)$ time.

The time complexity of the algorithm is $O(N \log N)$. In general, the number of viewpoints selected is not minimum and the results depend heavily on steps 1 and 2. [Note: One possible alternative is to combine steps 1 and 2 together by triangulating a set of line segments in $O(N \log N)$ time.] The viewpoints selected for two test scenes, which are shown in Figures 5-1 and 5-2, are illustrated in Figures 5-3 and 5-9 respectively.

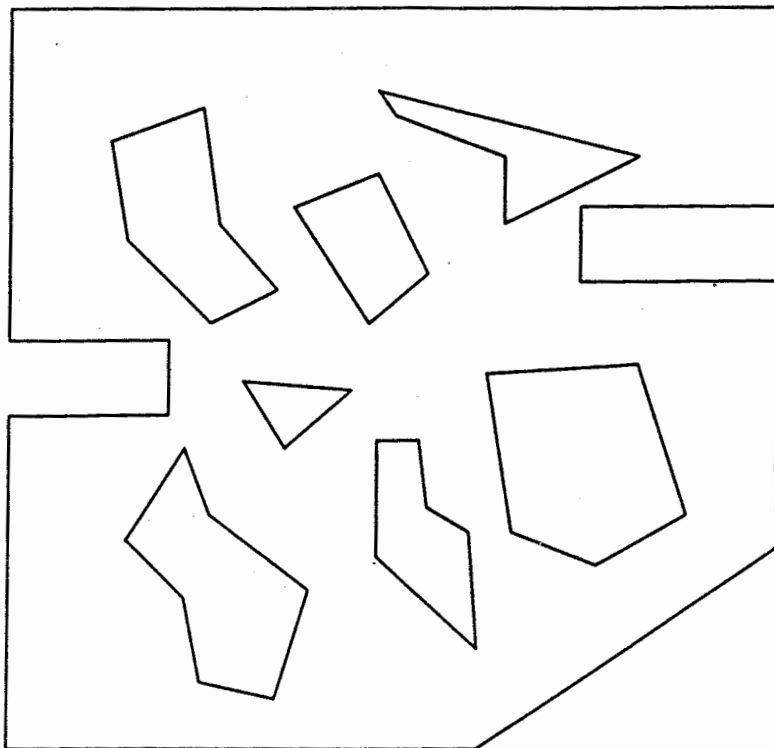


Figure 5-1: Scene 5-1: A test example for a cluttered scene.

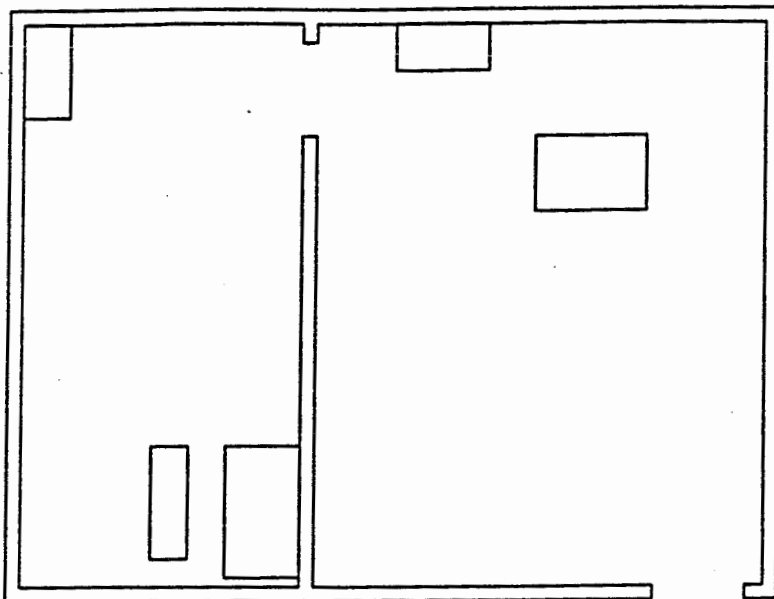


Figure 5-2: Scene 5-2: A more realistic scene.

5.2.2. Covering Scheme

Since visibility is a local property which is restricted within a free space and since, in general, more information than expected will be sensed from the sequentially selected viewpoints, a method of dynamic selection of viewpoints may reduce the total number of viewpoints. Accordingly, we have devised another approach for planning viewpoints. Instead of partitioning, this approach tries to cover the edges of a simple polygon which has been formed to represent the free spaces with star-shaped polygons.

The approach consists of the following steps:

1. As set out in substeps a to g, chain and order all the edges in $O(N^2 \log N)$ time, where N is the total number of vertices for the given set of objects and the environment.
 - a. Find the convex hull of the vertex set of the objects in $O(N \log N)$ time.
 - b. Keep those added edges which connect two different objects and remove those added edges which connect two different vertices of the same object in $O(N)$ time.
 - c. If a connecting edge intersects objects in the "outer ring" or the environment, replace the connecting edge by several connecting edges which form a path along the edges of the outer ring or the environment. The time complexity is $O(N \log N)$ in the worst case.
 - d. Order the objects along the ring; this will require $O(N)$ time in the worst case.
 - e. Ignore the edge which connects the first and last objects in the same ring and connect the vertex of the last object with the nearest vertex on the ring one level out or on the environment; the time complexity is $O(N)$ in the worst case.

- f. If there are some objects enclosed in the ring, then repeat from substep a. [Note: the inside edges of the objects in a ring and the connecting edges of the ring determine a new level of environment for the inside objects.]
- g. Choose the edge that connects the outermost ring with the environment as the first edge and order all edges by keeping the innermost of the objects and the environment on the left side of each edge. (See Figure 5-4). For each ring, first order the outer edges of the ring in an anti-clockwise order, then turn to the inside and order the inside edges of the ring in a clockwise order. During this process, when a connecting edge comes out from the end vertex of a traveled edge, the connecting edge will be traveled first except when it connects with the outer ring. The time complexity could be $O(N^2)$ in the worst case, as there could be $O(N^2)$ edges added.
2. Along the chain of edges, beginning from a real incompletely seen edge, use a greedy method to find the non-empty free intersection of the half-planes which are defined by the successor edges in $\text{MAX}\{O(hN), O(N \log N)\}$ time, where h is a constant, the maximum number of successor edges. [Note: Here the "greedy" method tries to observe as many successor edges as possible by finding the minimum non-empty intersection of half-planes in a valuable free area.] When only inner edges of a ring are considered, the free area between the inner edges of the ring and the outer edges of its inner ring is considered as valuable, otherwise the free area between the outer edges of the ring and the inner edges of its outer ring is considered as valuable. The determination of whether an intersection point locates in a free area requires $O(\log N)$ time for each query [25], while the preprocessing task runs in time at most $O(N \log N)$ [Note: The overlapping edges need not be considered more than once].
3. In combination with the previous step, determine a viewpoint in the intersection area from which at least the first edge to be considered should be completely seen. This can be done in $O(hN)$ time. This means that, when an inner edge of a ring is first considered, the outer edges of the inner ring or the other inner edges of this ring should not occlude it from the selected viewpoint. Thus for each candidate viewpoint, $O(N)$ time is needed.

4. The following $O(N \log N)$ approach is used to identify the edges of the objects and the environment which are completely seen from the current viewpoint:

- a. Move the origin of the coordinate system to the current viewpoint and change the coordinates into polar coordinates in $O(N)$ time.
- b. Order the vertices by their polar angles and put them in the list "SEENV" in $O(N \log N)$ time.
- c. Connect each vertex to the origin with a test line segment, label the test line segments with the edges incident from the vertex, and put all edges in the list "SEENE" in $O(N)$ time.
- d. For all of these test line segments report the nearest proper intersection points with the edges of the objects and the environment in $O(N \log N)$ time [11].
- e. If a test line segment has any intersection point on it, then its related vertex and edges are not seen or not completely seen, and they are deleted from the lists "SEENV" and "SEENE" respectively. This step requires $O(N)$ time in the worst case.
- f. Consider the remaining vertices in "SEENV" in turn; if the successor of a vertex belongs to the same edge, then the edge is completely seen; otherwise the edge, which is left-incident from the vertex, is not completely seen, and it is deleted from list "SEENE". This can be done in $O(N)$ time.
- g. Remove all the test line segments. The edges which remain in the list "SEENE" are completely seen from the current viewpoint.

5. Ignoring those edges which are already completely seen and the connecting edges in the edge chain, find the next unseen or partly seen edge in $O(N)$ time. If there is one, go to step 2. otherwise stop the procedure.

The total running time of the approach is $O(N^2 \log N)$ in the worst case. The viewpoints selected for the two test scenes 5-1 and 5-2 are shown in Figures 5-4 and 5-10 respectively.

5.3. Planning the Navigation Route

After the viewpoints have been selected, their order implies a corresponding navigation route, but the route is usually not optimal. For a patrol robot, it is desirable to reorder the sequence of the viewpoints and arrange the navigation route accordingly.

The procedure for planning the navigation route for the partition scheme is as follows:

1. Find the midpoint for each inserted edge; it requires $O(N)$ time in the worst case.
2. For each star-shaped polygon, connect the viewpoint in the polygon with each of the above midpoints by an edge which is a pass-way (i.e. an edge along which the mobile robot could pass); the pass-ways, midpoints and viewpoints together form a connected graph. Two viewpoints are said to be adjacent to each other, if they are adjacent to the same midpoint. The length of a pass-way between two adjacent viewpoints is the sum of the distances between the midpoint and the two viewpoints respectively. The time complexity is $O(N)$ in the worst case.
3. Using Floyd's algorithm [27], find the shortest Euclidean routes between all pairs of viewpoints in $O(VN^2)$ time, where V is the total number of viewpoints. The distance between two viewpoints is the length of the shortest route between them.
4. If a pass-way between two adjacent viewpoints is not the shortest route between them, then delete the pass-way from the formed graph, which requires $O(V^2)$ time in the worst case. The star-shaped polygonal decompositions and the graphs of pass-ways of the two test scenes in Page 86 are shown in Figures 5-5 and 5-11 respectively.
5. Ignoring the midpoints, the navigation problem is reduced to the "traveling salesman problem with triangle inequality" which is a well

known NP-hard problem in the strong sense. Christofides [20] has surveyed the exact and heuristic procedures for solving the "Traveling Salesman" problem. Using the Christofide's Heuristic algorithm [20], for all instances I , it is guaranteed to get a suboptimal solution $CH(I) < 3/2 * OPT(I)$ in $O(V^3)$ time.

Since V is less than N , the whole procedure requires $O(N^3)$ time in the worst case.

The procedure for planning the navigation route for the covering scheme is as follows:

1. Decompose the weakly simple polygon formed in Step 1.f of the covering scheme into convex polygons in $O(mN)$ time [81] [19] where m is the number of the reflex vertices of the weakly simple polygon.
2. Determine in which convex polygons the viewpoints are located in $O(N \log N)$ time [49].
3. If two or more viewpoints are located in the same convex polygon, then the distance between them is the Euclidean distance between them, and in the convex polygon the midpoint of an inserted edge is connected with its nearest viewpoint. If only one viewpoint is located in a convex polygon, then find the midpoint for each inserted edge and connect the viewpoint with each midpoint by a passway. If none are located in a convex polygon, then find the center of gravity of the convex polygon and connect it with the midpoint of each inserted edge of the convex polygon by a pass-way. It requires $O(m)$ time in the worst case, since the number of inserted edges is $O(m)$. The convex polygonal decompositions and the graphs of pass-ways of the two test scenes in Page 86 are shown in Figures 5-6 and 5-12 respectively.
4. This step is the same as steps 3 and 4 in approach 1.
5. Ignoring those centers of gravity, step 5 is the same as step 5 in approach 1.

Since m and V are less than N , the whole procedure requires $O(N^3)$ time in the worst case also.

5.4. Chapter Summary

Two 2-D algorithms have been devised to plan viewpoints for a patrol robot where there is prior knowledge of the objects and their positions. Although they do not offer optimal solutions, they do work in general situations and they are quite fast. Their solutions for the scenes shown in Page 86 are illustrated in Figures 5-3, 5-4, 5-9, and 5-10 respectively. For ease of comparison, the same object and edge ordering method has been adopted in both approaches. By elegantly ordering the objects and their edges, the complex relationships between objects are resolved by these algorithms.

The output of the above algorithms is post-processed by the corresponding navigation route planning algorithms. Their results are shown in Figures 5-7, 5-8, 5-13 and 5-14 respectively. These results show that the orderings of the viewpoints are reasonable and the corresponding navigation routes are feasible, but the local parts of the navigation routes could certainly be improved by some existing shortest path planning methods.

In combination with a movement control system, these approaches offer a good basis for a patrol robot working in a complex environment.

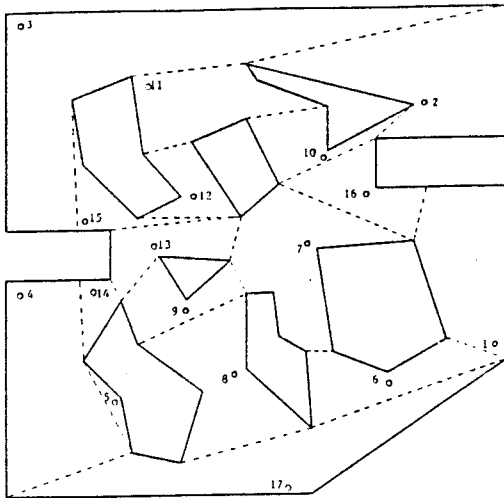


Figure 5-3: The selected viewpoints for Scene 5-1 using the partition scheme.

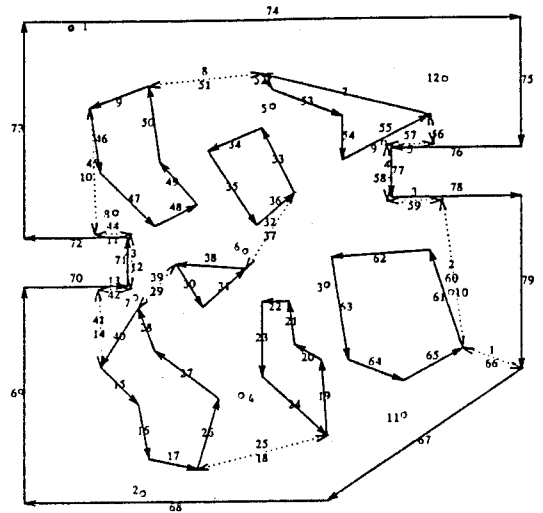


Figure 5-4: The selected viewpoints for Scene 5-1 using the covering scheme.

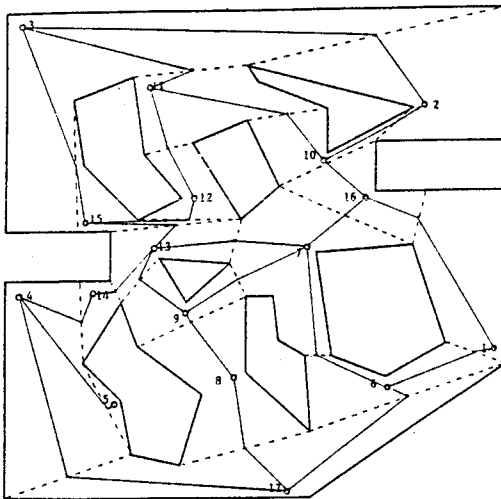


Figure 5-5: The star-shaped polygonal partition and the graph of passways for Scene 5-1.

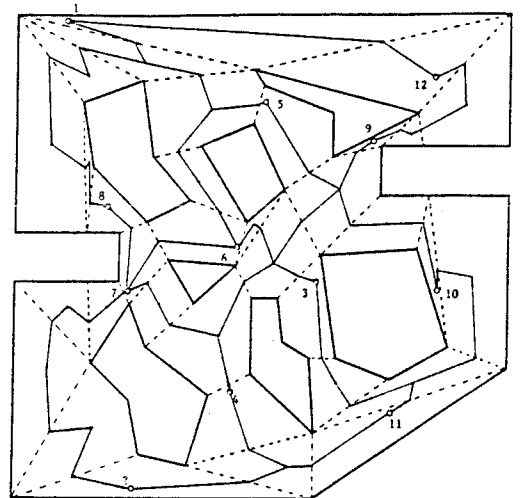


Figure 5-6: The convex polygonal partition and the graph of passways for Scene 5-1.

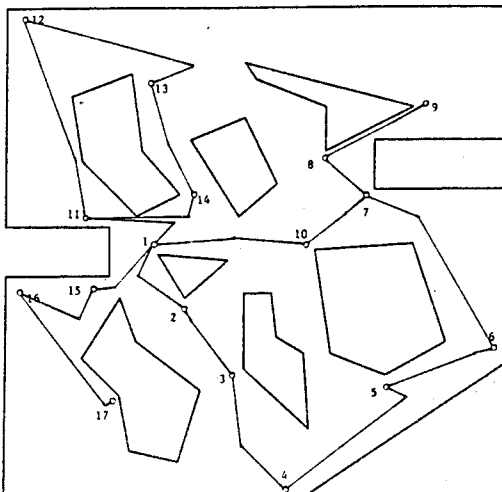


Figure 5-7: The reordered viewpoints and corresponding robot navigation route for Scene 5-1 using the partition scheme.

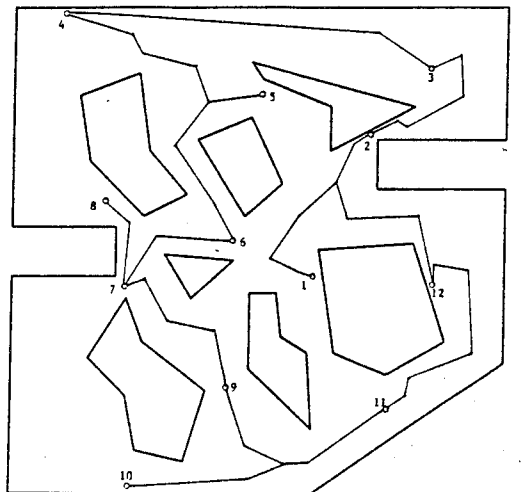


Figure 5-8: The reordered viewpoints and corresponding robot navigation route for Scene 5-1 using the covering scheme.

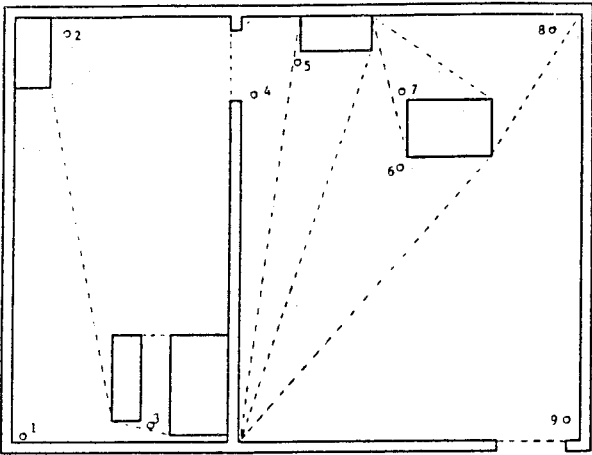


Figure 5-9: The selected viewpoints for Scene 5-2 using the partition scheme.

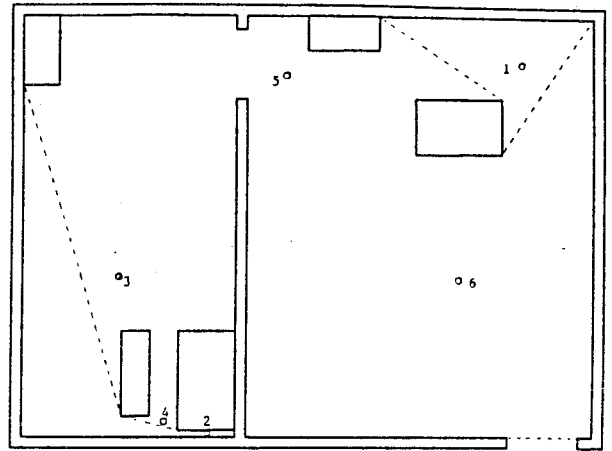


Figure 5-10: The selected viewpoints for Scene 5-2 using the covering scheme.

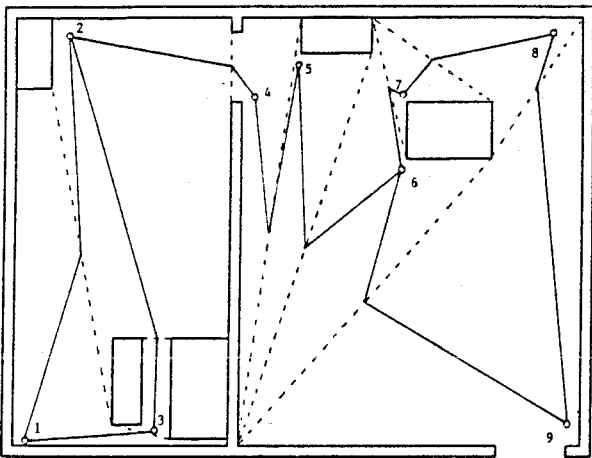


Figure 5-11: The star-shaped polygonal partition and the graph of passways for Scene 5-2.

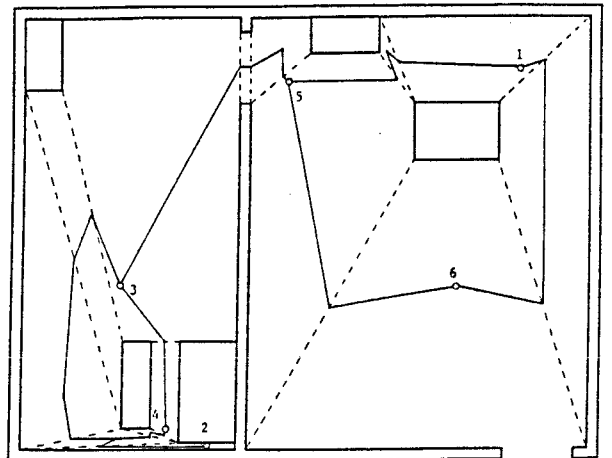


Figure 5-12: The convex polygonal partition and the graph of passways for Scene 5-2.

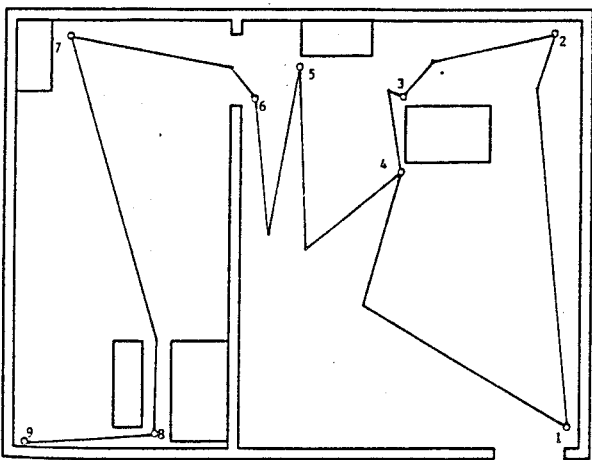


Figure 5-13: The reordered viewpoints and corresponding robot navigation route for Scene 5-2 using the partition scheme.

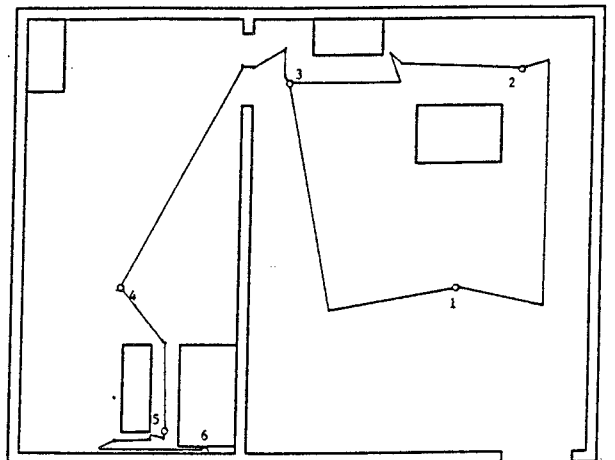


Figure 5-14: The reordered viewpoints and corresponding robot navigation route for Scene 5-2 using the covering scheme.

Chapter 6

Planning Views for the Incremental Construction of Body Models

6.1. Introduction

As stated before, a mobile robot must often work in a situation where the shape and dimensions of the environment are known, but where the bodies within the environment are unknown. In order to work within this environment the robot needs to incrementally construct models for the bodies. To do this it is important to select optimal viewpoints which allow the 3-D models to be constructed as efficiently as possible.

The factors which affect the selection of viewpoints include:

1. the properties of the bodies and their surrounding environment(e.g. complexity of body shape, self-occlusion, occlusion and the arrangement of empty spaces),
2. previously accumulated information on the bodies and the environment,
3. the movement capability of the robot (e.g. whether the robot can climb over or creep underneath the bodies and the location of the sensor on the robot's body), and
4. sensor characteristics (e.g. the size of the viewport, whether the sensor needs to be focussed, and the depth of field for the lens).

This chapter describes a new approach to the determination of the viewpoints when the body models are not known *a priori*.

6.2. General Approach

Our general approach is to first simplify the 3-D decision making problem into a 2-D problem by projecting the partial body models onto the horizontal plane and approximating the projections with line segments. After making this simplification, viewpoints are chosen to best resolve the general ambiguities in the scene. As these ambiguities are resolved, local details can be elaborated from successive viewpoints planned with local geometrical hints.

According to Chapter 4, as partial body models are constructed, they are projected to the x-y plane and a map formed by line segments can be built up. These partial body models and the map will be modified as new information is collected from successive views. The robot work environment (which is known in advance), forms the limited area within which the robot can move around. At a given point in the data gathering process, the previously accumulated information for objects indicates which areas in the map are free, which are occupied and which are unknown. The unknown areas are bounded by body outlines and Projected Viewlines (PVL) which indicate where further exploration should take place. Also they determine the directions for new views.

In the simplified 2-D situation, the lines in the map are ordered; the Projected Viewlines are put in a list and considered in turn. A PVL passing through a visible boundary point of an object and the extension of the corresponding visible

edge of the object indicate the two boundary edges of the area within which a new viewpoint should be located. An area for the location of a new viewpoint can be found by approximating the object with a square and by using the observed free area information. The currently known maximum dimensions of a body are used to estimate the size of viewport and the distance between the sensor and the body.

6.3. Planning Viewpoints in 2-D for Unknown Objects

In Chapter 5, the Partition Scheme selects the viewpoints statically and it cannot be extended to the case where only partial models are known.

We now describe an $O(N^2 \log N)$ scheme for planning viewpoints in the 2-D case where knowledge of the objects does not exist. As with the covering scheme in Chapter 5, this scheme also successively selects the viewpoints, but it tries to cover unoccupied spaces with star-shaped polygons. Note first that no matter where the first viewpoint is located in the environment, the result of the first view will be a star-shaped polygon which is made up of the visible object line segments and some PVLs which connect the visible line segments. The PVLs are not physical edges of the scene but each forms an edge to an unknown area located outside the polygon. The scheme consists of the following steps:

1. Order the edges of the star-shaped polygon in an anti-clockwise direction and put its PVLs in a list Q in $O(N)$ time, where N is the total number of edges of the objects and the environment. In the list Q , the information of the starting edge, which is connected with the starting point of a PVL, and the ending edge which is connected with the end point of the PVL, is also stored with the PVL. To discover those unknown areas corresponding to the PVLs of the star-shaped

polygon, the positions of viewpoints will be chosen in the related triangular free areas which are defined by the starting point and ending edge of the PVL.

2. The vertices are distinguished as the following 3 types: a false vertex which is the end point of a PVL, an exploring vertex which is the starting point of a PVL, and a known vertex that the close vicinities of its two emanating edges are at least viewed. All vertices of the star-shaped polygon, except the false vertices, are ordered first according to their X coordinates, then, according to their Y coordinates, for those having equal X coordinates; the ordered vertices are put into a list EV. This requires $O(N \log N)$ time.
3. The PVLs from the list Q are then considered in turn to resolve the ambiguities in the unknown areas. This involves the following substeps:
 - a. Assume that the first object to be explored is enclosed in a square defined by two edges, one of which is the visible object edge adjacent to the PVL under consideration (i.e., starting edge) and the other is a line orthogonal to the first starting at the vertex shared by the object edge and the PVL. The side length of the square is adopted as the estimated maximum dimension of the object.
 - b. If possible, the new viewpoint should be placed in the related free triangular area from which the two unseen edges of the square can be viewed; otherwise, the viewpoint is placed nearby the end point of the PVL in the free area from which as much as possible of the unknown part of the square can be seen. This requires a constant time.
 - c. For each selected viewpoint, the result is a new star-shaped polygon for which the edges are ordered in an anti-clockwise direction. The vertices of the new star-shaped polygon are also ordered and merged into the list EV. In the meantime, the PVLs corresponding to the new unknown areas are identified; these PVLs must start from (1) a new exploring vertex or (2) an old exploring vertex in EV from which an old PVL is at an angle to the known edge smaller a constant degree than the new PVL. The corresponding old PVLs are deleted from the list Q. The new PVLs which have been ordered are added to the end of the list

Q. This requires $O(N \log N)$ time. For these new unknown areas, new viewpoints will be selected in the new corresponding triangular free areas.

4. When an unknown area is discovered, the corresponding PVL is deleted from the list Q. Also, any old PVL will be deleted from the list Q, if its starting and ending points are on the newly perceived star-shaped polygon [note: this can be tested in $O(N \log N)$ time, since for each old PVL, its starting and ending points can be tested in $O(\log N)$ time.] and it does not intersect any new PVL [note: this can be tested in $O(N \cdot B_i)$ time, where B_i is the number of new PVLs of the new star-shaped polygon, since for each old PVL the test can be done in $O(B_i)$ time.]. The number of total PVLs $\sum_i B_i$ is $O(N)$, since each vertex of the objects only can cause at most a constant number of PVLs. The process goes to step 3 and continues until the list Q is empty.

Solutions obtained with this approach for the scenes shown in Figures 5-1 and 5-2 are illustrated in Figures 6-1 and 6-2 respectively. The starting point of the robot is at position 1. The solution in Figure 6-2 may further refer to Figures 4-4 to 4-9 in Chapter 4. If the view directions of the robot are restricted to the PVLs which correspond to the unknown areas under consideration, then a few more viewpoints are required, since some old PVLs in free areas which are no longer covered by the separating star-shaped polygons may not be deleted at step 4.

The covering star-shaped polygons form a tree structure: each node represents the viewpoint of a star-shaped polygon; a node has those viewpoints, which explore the PVLs in its corresponding star-shaped polygon, as its daughter nodes; a branch between a node and its daughter node represents the straight-line route between them. Thus a simple method to find the route between two viewpoints is to find their common ancestor in the tree structure. The robot could go from the first

viewpoint to the ancestor then to the second viewpoint. For example, an exploration route for the scene shown in Figure 6-2 could be line segments: 1 -> 2 -> 1 -> 3 -> 1 -> 4 -> 1 -> 2 -> 5 -> 2 -> 1 -> 4 -> 6 (refer to Figure 4-9). Considering that the planned viewpoints 4, 5 and 6 are in the second room and that the boundary(door) between the first and second rooms is unobstructed, the returns to the first room are unnecessary. Thus an improved exploration route can be arranged as shown in Figure 6-3. Using a more sophisticated segmentation method which divides the free area into sub-areas, a better exploration route could be found, but it would certainly require more time for computation.

6.4. Planning Viewpoints for 3-D Local Ambiguities

After the global scene has been reconstructed from the viewpoints planned by the 2-D method described above there will generally still be some local ambiguities since most bodies have features in the vertical plane.

The system that we have described in Chapter 3 for incrementally constructing body models will identify the environment, decompose the scene into bodies, and indicate the body parts where further geometric information is required. For these local ambiguities, the corresponding local geometrical properties of a body, such as junction types from certain views and surface categories, offer hints for planning the new viewpoints. These kind of hints can be organized into an expert system which helps the viewpoint planning system to make appropriate decisions to resolve the local ambiguities. The rules for planning new viewpoints in 3-D to find local ambiguities are set out in detail in Appendix H.

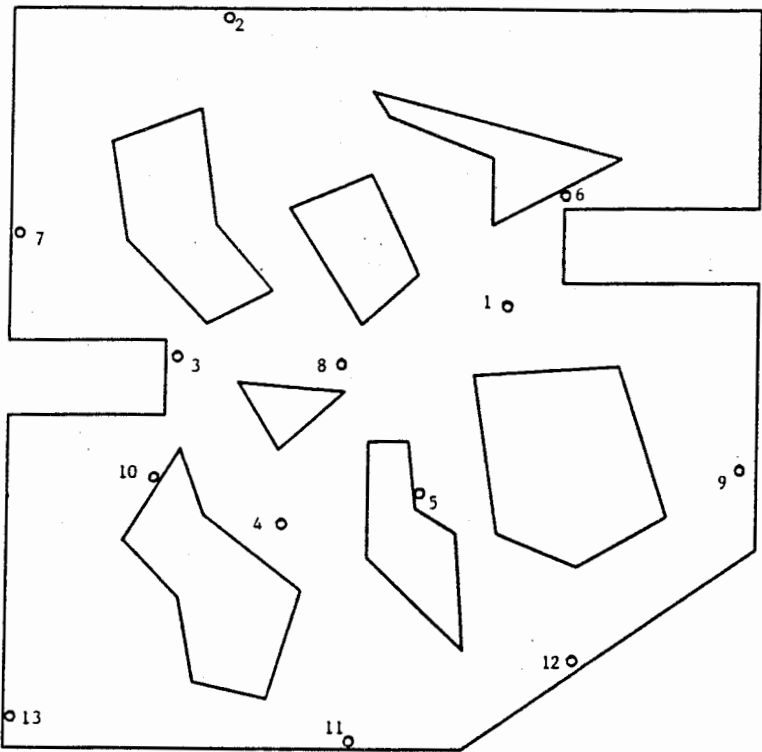


Figure 6-1: A solution for Scene 5-1 where no prior knowledge is assumed for the objects.

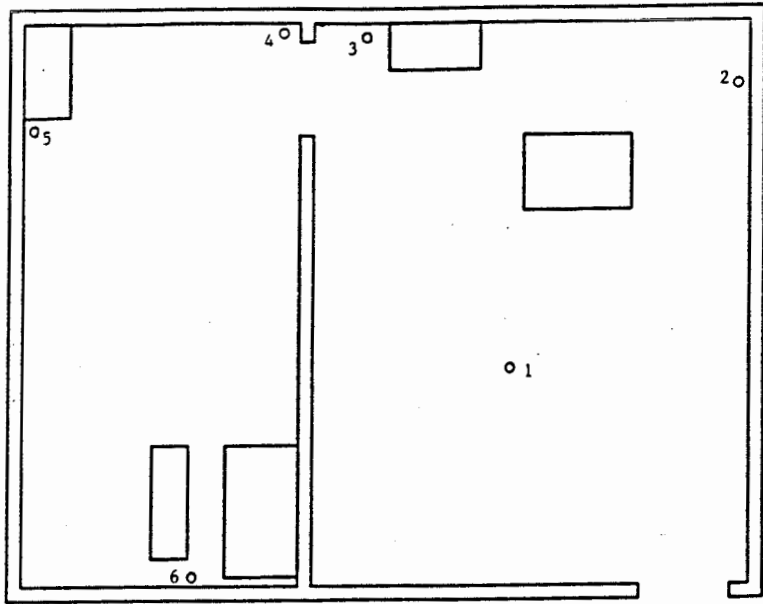
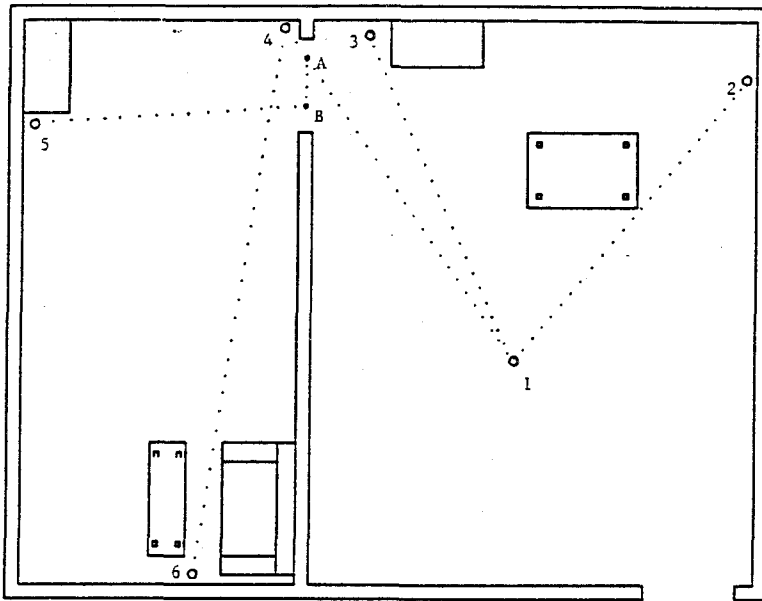


Figure 6-2: A solution for Scene 5-2 where no prior knowledge is assumed for the objects.



The exploration route is line segments: 1→2→1→3→1→4→A→B→5→B→A→4→6.

Figure 6-3: A solution for Scene 5-2 with an improved exploration route for the robot.

Since the horizontal projections which are separated polygons correspond to the bodies or body groups that do not touch each other and since they can easily be ordered (e.g., by the method described in Chapter 5), it is clear that the viewpoint planning system can consider the bodies or body groups in turn. For a separated body or a separated body group which has few (e.g. less than 5) local ambiguities, the planning system will consult the expert system and use the "greedy" method to find some common viewpoints to solve the ambiguities. For a separated body group which has more local ambiguities, the planning system will consider another analyzing level of viewpoint plan in such situations as several objects are on a table; or will queue ambiguities in a list and solve them in turn.

When a viewpoint has been chosen by the planner, a new view will be taken.

The sensory information which has been collected from the viewpoint will be matched with the constructed body models and the new information will be added into the models. New ambiguities will also be identified. This procedure will be repeated until all ambiguities have been resolved or until it is determined that they cannot be resolved because of the limits on robot movement.

6.5. Chapter Summary

A new approach has been devised for determining viewpoints in a known environment where the goal is to incrementally construct body models. This approach works well for a variety of scenes; a typical example is shown in Figure 1-1. In general, it offers good and efficient solutions for scenes where the bodies are well scattered in complex environments. Thus, the occlusion and self-occlusion problem in scene analysis may be resolved by a system such as that described above. In combination with a trajectory planning system and a model constructing system, this approach offers a good basis for a flexible mobile robot working in complex environments.

Chapter 7

CSG-EESI: A New Solid Representation Scheme and a Conversion Expert System

7.1. Introduction

Methods for the representation of solids are important in computer vision, computer graphics, CAD and related areas. In scene analysis, if the exact models of bodies in a scene are known *a priori*, then the corresponding recognition process can be relatively easy. This can be based on such characteristics [99, 76, 67] as: (1) the general features of the models (e.g., size, number of faces, edges or vertices, connections between faces), (2) the characteristic features of the models (e.g., holes, special vertices or their combinations), and (3) a set of their 2-D projections. If the exact models for bodies in the working environment are not known in advance, then recognition must be based on the conceptual models for those classes of objects which are expected to be found. For example, in a domestic environment it is generally expected that there are sofas, chairs, tables etc. Could a robot recognize these bodies, as a person does, when it enters a new room? Could the robot execute a command such as "find those chairs which have broken legs in the second office on the right."? Many characteristics of bodies are used by humans in order to classify them. These include:

1. geometric properties, such as structure, dimensions, shapes, characteristic features, and location in an environment;
2. physical properties, such as lightness, colour, texture, weight, and mobility;
3. frequency of occurrence;
4. usage and other functional characteristics.

Among these, the geometric properties are the most important, stable and frequently used; the most valuable geometric properties for the recognition of common complex objects, such as desks, tables and chairs, is the structural information. Humans can easily decompose a body into its subparts and get the structural information, even when there are no hints like colour, texture, material, environment and functional characteristics, though such decompositions may not be unique. Could a robot do this automatically? In previous chapters, we have described the methodology and techniques used in a vision system for a mobile intelligent robot. This vision system explores the environment and constructs 3-D BR-like models of the bodies in an indoor scene using automatically selected views. Thus we here assume that the complete BR representation of a body is known. The purpose of this chapter is to describe a way to convert this Boundary Representation into a useful CSG like representation where structure is explicit.

In solving the decomposition problem, the hard questions which must be answered include:

1. What kinds of parts can be viewed as simple and primitive?
2. What kinds of decompositions can be considered reasonable?

3. What kinds of geometric information can be used as cues for the decomposition?

A solution to these problems is presented in this chapter, but we limit ourselves to bodies formed by planes and quadric faces. This includes many of the solids found in practical robot vision situations. In the next section, we present a new representation scheme for describing 3-D mechanical parts and structured bodies. Since the new method combines features of the Constructive Solid Geometry (CSG) representation and an extension of the Enhanced Spherical Image representation (ESI), we designate the method CSG-EESI. In this scheme the body model can be roughly divided into two levels. The higher level corresponds to a restricted CSG tree which contains the structural information describing how the various subparts form the body. The lower level contains the geometric information for those simple subparts and represents them by an extension of Enhanced Spherical Images. The CSG-EESI scheme may be used both as the medium between pictorial models and relational models and as an internal model to facilitate the recognition of bodies.

The critical need for a conversion method from BR-like models to the CSG-like model is revealed in the third section. In the fourth section, we describe a hierarchical expert system which converts BR-like models into the CSG-EESI representation. On the basis of the type and convexity of faces and the convexity of their intersection edges, the system deduces reasonable decompositions for bodies and offers a structural description. This expert system was developed in C-PROLOG on a VAX 11/750. Two groups of test bodies and their results are

given in section five. The expert system partly fulfills the conversion gap from BR to CSG representations.

7.2. The CSG-EESI Representation

The CSG-EESI scheme is a combination of the CSG scheme and an extended version of the ESI scheme. In general, the ESI scheme developed by Smith is only suitable for the representation of convex polyhedra. In order to avoid this restriction we have developed an Extended ESI (EESI) scheme which can represent the primitive bodies formed by planar faces and at most one convex quadric face (or a half-torus). These primitive bodies do not need to be convex.

Usually, a Boundary Representation of a body explicitly represents the faces, edges and vertices as well as the relationships between them. This complete information is quite useful for graphics and computer vision. It also allows Boundary Representations to represent complex bodies. Although this complete information includes a lot of redundant information, it does not explicitly offer the structural information. The ESI (or SI) scheme discards the redundant information of an explicit Boundary Representation, such as edges, vertices and the adjacency relationships between faces. Further, it puts the center of mass of a body at the center of the Gaussian sphere, then discards the position data of the body center. This makes ESI (or SI) concise and allows an ESI (or SI) model to rotate with the body it represents. But, the limited information offered by ESI (or SI) restricts its representation domain and its usefulness.

To enlarge the representation domain of ESI, the other intrinsic information of a

body needs to be added. Although adjacency relationships between faces could be discarded for convex bodies, for non-convex bodies, the adjacency relationships at concave edges should be retained. ESI satisfies the condition of the center of mass, and based on Minkowski's theorem the perpendicular distances between the faces and the center of the Gaussian sphere could be inferred from the information on areas and the directions of normals [53]. But, for non-planar bodies the location of a center of mass is hard to identify. Thus, a better way seems to be to retain the information of perpendicular distances between faces and the selected origin. This also makes the construction of a body from its representation easier. In the domain of quadric and planar bodies, quadric surfaces have their intrinsic information, such as surface types, symmetric centers, symmetric axes and other parameters. This property allows quadric faces to be represented in quite a concise way.

Based on these considerations, the EESI representation has been defined. An EESI model consists of a set of attributed vectors with unit length. The indicator, direction and scalar parameters of a vector are chosen as shown in Table 7-1. The indicator identifies which kind of surfaces or which symmetric axis of a quadric surface the vector represents. Some vectors are not independent and are linked together as a group. This occurs only when a group of planes is connected by concave edges or when more than one symmetric axes are needed to specify for a certain quadric surface (e.g. ellipsoidal surfaces, or a half-torus). If a primitive body has a quadric face, then the location of the origin of the local coordinate system for the body is fixed to the symmetric center of the quadric

Surface	Indicator	Direction of vector	Scalar parameters	Linked vectors
Plane	pla	Plane normal direction	Directed distance between coordinate system origin and plane	Concave connected planes
Circular Cylinder	cyl	Direction of cylinder axis	Diameter of the cylinder	
Elliptic Cylinder	ecyla	Direction of cylinder axis		axes
	ecylb	Direction of major axis of ellipse	Long diameter of the ellipse	axes
	ecylc	Direction of minor axis of ellipse	Short diameter of the ellipse	axes
Sphere	sph		Diameter of the sphere	
Circular Cone	con	Direction of cone axis	Angle of cone	
Elliptic Cone	econa	Direction of cone axis		axes
	econb	Direction of major axis of ellipse ¹	Long diameter of the ellipse	axes
	econc	Direction of minor axis of ellipse	Short diameter of the ellipse	axes
Ellipsoid	ipsa	Direction of the principal axis	Long diameter	axes
	ipsb	Direction of the second axis	Second long diameter	axes
	ipsc	Direction of the third axis	Short diameter	axes
Half-Torus	hta	Direction of the principal axis	Revolution diameter	axes
	htb	Direction of the symmetric axis	Diameter of its circular section	axes
Right-Half-Plane (RHP) ²	rhp	Plane normal direction	Directed distance between coordinate system origin and plane	Concave connected planes
Left-Half-Plane (LHP)	lhp	Plane normal direction	Directed distance between coordinate system origin and plane	Concave connected planes

Table 7-1

Vector components in the EESI scheme.

¹The distance between the apex and the ellipse of the elliptic cone is 1.

²The projection of the principal axis of the considered half-torus divides the plane into RHP and LHP.

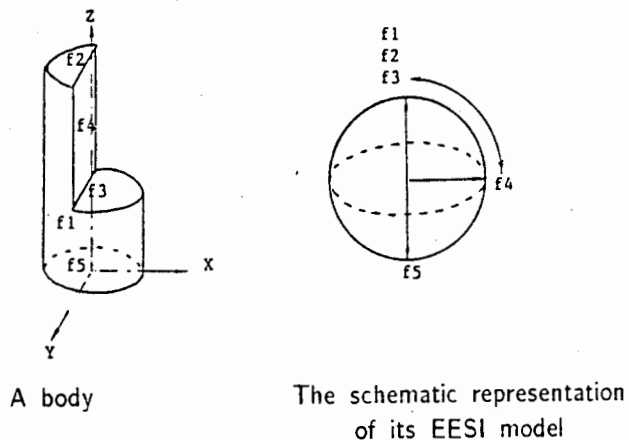
surface as shown in Table 7-2. An EESI model is allowed to contain a group of linked vectors which represents a convex quadric surface, but no more such groups¹⁵. If an EESI model contains a group of linked vectors which represents a convex quadric surface, the planar faces represented by the other vectors must locate in the extension of the quadric surface. In general, the EESI scheme can represent (1) a convex body formed by planar faces, (2) a convex body formed by both a convex quadric face and a number of planar faces which are in the extension of the quadric face, (3) a body formed by a half-torus and a number of truncating planes or half-planes, or (4) a body with some externally visible planar holes and/or concave cuts which is made by removing convex planar shapes from the previous three kinds of bodies. Thus, different EESI models reflect the shape differences between relatively simple bodies which contain the minimal structural information. Two examples of EESI models are shown in Figure 7-1.

The vectors, which represent the cylindrical, ellipsoidal and toroidal surfaces, may take the reverse direction. In a strict sense, EESI representations are not unique, but this non-uniqueness is limited to a rotation and a translation. In other words, for any body, the vector directions in its EESI model are unique up to a rotation of the coordinate system and the distances between planar faces and the center of the coordinate system are unique up to a translation of the center.

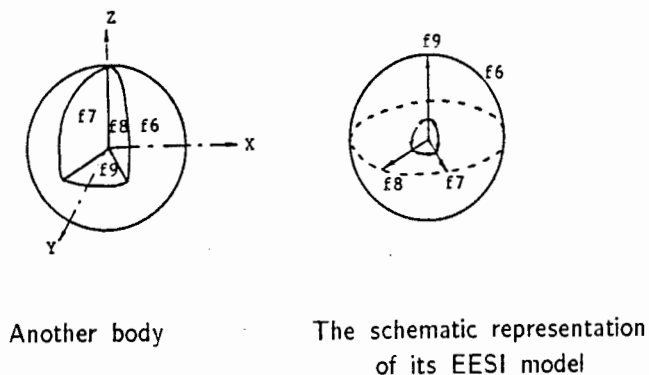
¹⁵In general, the connection between two quadric faces can not be simply distinguished as the convex-connected or the concave-connected mode, and a combination of two quadric faces can not be implicitly represented by the simple syntactic format of the EESI scheme.

Type of surface	Origin location of local coordinate system
Plane	Option
Circular Cylinder	Along axis
Elliptic Cylinder	Along axis
Sphere	Center
Circular Cone	Apex
Elliptic Cone	Apex
Ellipsoid	Center
Half-Torus	Center of torus

Table 7-2: The location of the origin of the local coordinate system for a primitive body.



Name	Indicator	Direction	Parameter	Linked List
vector(f1.	cyl.	(0. 0).	10.	—).
vector(f2.	pla.	(0. 0).	20.	—).
vector(f3.	pla.	(0. 0).	8.	(f4)).
vector(f4.	pla.	(90. 0).	7.	(f3)).
vector(f5.	pla.	(180. 0).	0.	—).



Name	Indicator	Direction	Parameter	Linked List
vector(f6.	sph.	—.	30.	—).
vector(f7.	pla.	(90. 45).	0.	(f8. f9)).
vector(f8.	pla.	(90. 135).	0.	(f7. f9)).
vector(f9.	pla.	(0. 0).	0.	(f7. f8)).

Figure 7-1: Two examples of one kind of formal representation for the EESI model.

An EESI model can be viewed as a restricted tree of a CSG model where the primitives are half-spaces and a complete quadric surface. Between the linked planes, union operators are applied first to form the concave cuts or holes of the body. The half-torus can be viewed as the intersection of a torus with a half-space; and a half-plane can be viewed as a $3/4$ space which is the union of two half-spaces. The intersection operators are then used between the surfaces and the constructed subparts to further construct the primitive body. Thus the EESI scheme inherits properties from both the ESI and CSG schemes. As in ESI, an EESI model is not affected by translation of the body; rotation of the body induces a rotation of the EESI model. As in CSG, the EESI scheme is unambiguous. CSG trees based on general half-spaces may represent unbounded sets and therefore are invalid, thus the validity of a EESI model relies on the constructor. Since EESI is a concise representation, it is much easier to check its validity than in Boundary Representations. The special thing here is that an EESI model has its own surface representation and shelters a primitive body from a top-level CSG tree which is a volumetric representation.

CSG-EESI representations are binary trees. Nonterminal nodes represent operators which may be either rigid motions (translations or rotations), union, or regularized difference; terminal nodes are either primitive bodies which are represented by EESI models, or transformation leaves which contain the defining arguments of translations or rotations. Unlike the simple CSG scheme, the CSG-EESI has an infinite number of primitives but restricted binary trees. If a body has concave quadric faces, then the regularized difference operators are applied

between the related subparts, otherwise, the union operators are applied to combine the subparts together.

The CSG-EESI representation includes relatively little redundant data, so that it is convenient for storage and ease of creation. The CSG-EESI scheme is potentially capable of covering a domain as rich as those CSG schemes whose primitives are half-spaces, spheres, circular cylinders, circular cones, ellipsoids and toruses. Many mechanical parts and other man-made objects belong to this domain. This scheme roughly divides a body model into two levels. The higher level corresponds to a restricted CSG tree which contains the structural information describing how the various subparts are combined to form the body. The lower level uses EESI models to describe the geometric information of those simple subparts.

The CSG-EESI representation is unambiguous. Since any CSG tree is a valid representation of an r -set if its primitive leaves are valid, it follows that a CSG-EESI model is valid, if its EESI primitives are valid. CSG-EESI representations are not unique. Just as different people may decompose the same body into different combinations of components, a body may be represented by different CSG-EESI representations which are equivalent. By using an automatic conversion system, such as the system described below, which uses the same rule base to convert both the prototype bodies and the perceived bodies, the variety of CSG-EESI representations for a body will be limited to a minimum number.

The CSG-EESI scheme has the following advantages:

1. Since the CSG tree is built on EESI representations, the variety in the tree will be limited; the structure of the CSG tree lays emphasis on the structure between parts.
2. The model of a body after rotation is a binary tree which has the rotation operator as its root node, the original CSG-EESI model as its left-hand branch, and the arguments of rotation as its right leaf. The rotation node can also be absorbed into the original CSG-EESI representation. After rotation, the union and regularized difference operators in the CSG tree do not change; the transformation leaves for the motion operators can be easily derived from the arguments of the rotation; EESI models are rotated just like a sphere. Thus the model of a body after rotation can be easily derived from the original.
3. If methods for approximately determining the axes and calculating the smallest including boxes for EESI models could be developed, the derivation of higher level relational models from the CSG-EESI representations would be straightforward.
4. As with engineering drawings, conventional tolerances on distances and dimensional parameters can be included in the EESI model. Conventional tolerance limits can also be included in the transformation leaves of a CSG tree. This tolerance specification satisfies the proposition, proposed by Requicha [74], that such specifications consist of: (a) an unambiguous representation for a nominal solid S ; (b) a representation for a decomposition of the boundary of S into subsets F_i , called nominal surface features, which are homogeneously two-dimensional and whose union is the boundary of S ; and (c) a collection of geometric assertions A_{ij} about S 's nominal surface features. Recently a sophisticated scheme which is based on a tolerance theory and associates tolerances with CSG models has been developed by Requicha and Chan [75]. Thus the CSG-EESI scheme could be extended to handle tolerances. But, in the strict sense, after adding in the tolerance information, a CSG-EESI model becomes ambiguous. The validity of tolerance specifications is largely an open problem [73].
5. For the restricted domain which we are considering, the CSG-EESI models can be derived from BR-like models or 2 1/2 D sketches.

7.3. Conversion from BR to CSG-EESI

The CSG-EESI scheme is designed as an internal representation for computer vision, although it should also be useful in computer graphics, CAD and related areas. Usually general vision systems involve many levels of representation which correspond to different levels of processing. At the lowest level are the sensory images, and at the highest level are the symbolic descriptions of objects which can be mapped into class descriptions and the relationships between those objects. Between these two extremes, there may be several intermediate levels of representation. In this case, we are concerned with deriving CSG-EESI models from 2 1/2 D sketches and deriving higher level representations such as relational models from CSG-EESI models. Since the derivation of a boundary representation from 2 1/2 D sketches has been described in Chapter 3, one important step is to look for a conversion method to derive the CSG-EESI model from BR-like models.

In computer graphics and CAD both CSG and BR methods are widely used. For CAD, Wesley and Markowsky [103] further pointed out that there are many existing nonsolid design systems and data bases such as paper based 2-D engineering drawings, computer based 2-D engineering drawings and computer based 3-D wire frames, and that there is a need to provide tools for the conversion of non-solid forms to solid forms such as BR or CSG. At this time, there are algorithms which convert cell decompositions, spatial enumerations, simple sweeps and CSG representations into boundary representations [72]. Although Vossler [100] has developed an algorithm to convert sweeping models to CSG models which uses blocks, right wedges and cylinders as primitives, the domain of

sweeps is much smaller than the domain of Boundary Representations or the domain of CSG representations. Vossler's algorithm searches the input sweep outline until a two-dimensional pattern of one of the simple sweep solids is found. The simple sweep solid is then stored on a stack along with a Boolean operator, and the complexity of the outline is reduced. The search and outline reduction process continues until the outline no longer encloses any area. At that time, the CSG model is formed by replacing the simple solids with primitives. This algorithm is essentially two-dimensional and it is hard to extend it to three-dimensions. Thus a critical need still exists for a conversion method to derive the CSG-like model (such as CSG-EESI) from BR-like models.

In this chapter, the task of deriving a CSG-EESI model from BR-like models is approached by using an expert system to decompose bodies. According to the types and convexities of faces, the convexities of intersection edges and the connecting relationships among the faces, the expert system recursively decomposes a body into its sub-parts until each of them is represented by an EESI model. In this process a CSG tree which stands on the top of the decomposed sub-parts is formed. Once a simple part has been found, its redundant information is omitted and the necessary intrinsic information of its faces is derived and added. Based on these, EESI models of the sub-parts are constructed.

In general, bodies have considerable variance. Even in our limited domain, they also have different types of surfaces and connectivities and geometric characteristics. This suggests that a well-built hierarchical expert system is more suitable for our problem.

7.4. The Conversion Expert System

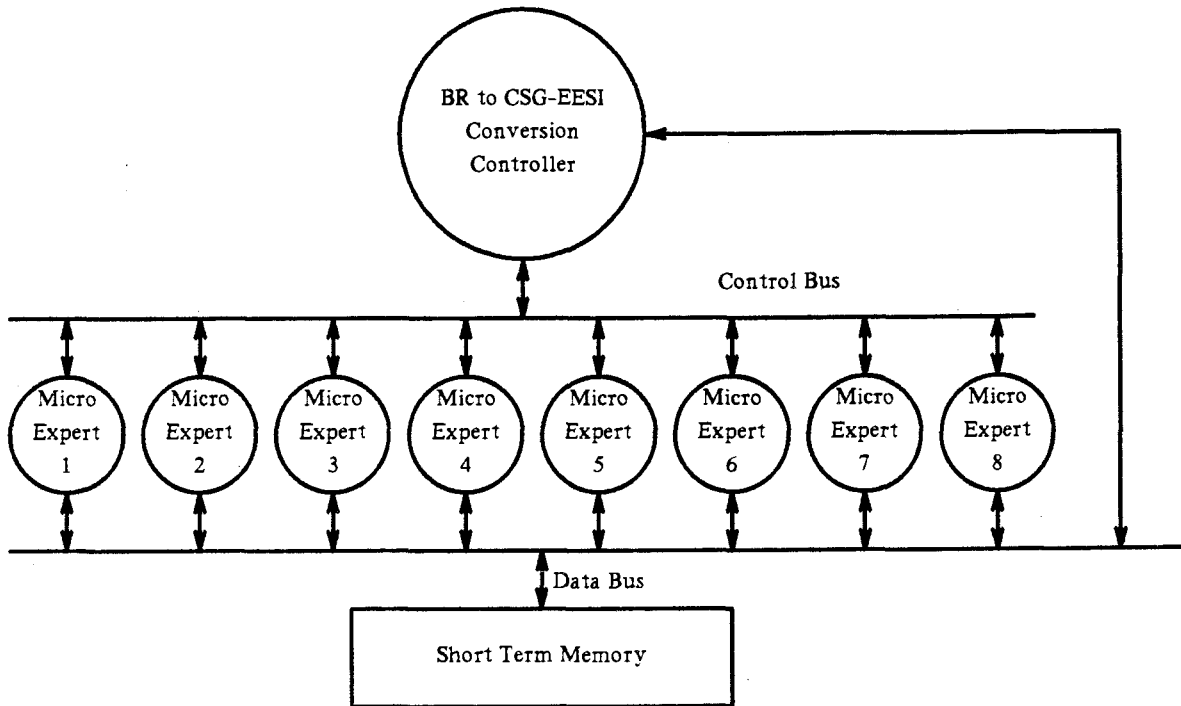


Figure 7-2: The schematic sketch of the conversion expert system.

In the CSG-EESI conversion system which has been developed, there is an expert system called the "BR to CSG-EESI Conversion Controller" and 8 micro-expert systems. This is illustrated schematically in Figure 7-2. In this system knowledge is represented as a general pattern-invoked program and as first order predicate logic. More precisely, the Controller distinguishes different situations and specifies the tasks in order to first deal with the parts containing concave quadric faces, then the parts containing convex quadric faces, and at last the planar parts.

In considering a part, it selects a primary face which is either a quadric face or a plane which is concave-connected by a set of other plane groups. According to the type and the convexity of the primary face and the types of connection between the primary face and other faces, the Controller invokes the appropriate micro-expert systems to decompose the part under consideration.

The micro-expert systems perform the following tasks:

1. According to the types of connection between the primary face and other faces, select a suitable location where a separating plane can be placed to reasonably separate a body into two subparts, and calculate the parameters and connectivities of the separating plane. The reasonableness of the separation is based on: (a) Whether the separation occurs at or begins from some concave connected places. (b) Whether after separation faces remain as complete as possible. Thus the result of a body decomposed by the expert system is often similar to (a) a manufacturing process for the body, (b) the result of break of the body by natural forces, or (c) a decomposition made by a human.
2. Decompose the body into two parts and update the consistencies and connectivities of these two parts.
3. Form a branch of the CSG tree to represent the decomposition.

When a micro-expert system has finished its work, it communicates with the Controller. According to the results of this decomposition step, the Controller will form the new task and assign it to the appropriate micro-expert system.

The knowledge and rules stored in the Controller and micro-expert systems are set out in detail in Appendix I. They are based on the geometric properties of faces and the geometric constraints between faces; thus they are reliable and

stable. Since they only use geometric cues, from the point of view of a human, the decomposition results sometimes seem conservative. The main difference between the rules in the micro-expert systems is the method used to select a suitable position to place the separating face on the basis of the intersection types and the intersection point sets between the primary face and its related faces.

Figure 7-3 shows an example of a decomposition made by the BR to CSG-EESI Conversion Expert System. The Controller of the CSG-EESI conversion system assigned the decomposition task to Micro_expert 1 according to Controller Rule 1.a., after it found that the body shown in level 1 of Figure 7-3 has a concave cylindrical face. According to Micro_expert 1, the body was analyzed as the difference of two parts: Part 2L and Part 2R, which are shown in level 2 of Figure 7-3. When the control returned to the Controller, according to Controller Rule 2.a., it assigned the decomposition task to Micro_expert 4 in order to deal with the bottom cylinder. According to Micro_expert 4 Rule 3, Part 2L was analyzed as the union of two parts: Part 3L and Part 3R, which are shown in level 3 of Figure 7-3. For Part 3R, the Controller assigned Micro_expert 4 to deal with it. According to Micro_expert 4 Rule 2, it forms an EESI model. For Part 3L, the Controller assigned Micro_expert 4 to further deal with the middle shift. According to Micro_expert 4 Rule 1, Part 3L was analyzed as the union of two parts: Part 4L and Part 4R, which are shown in level 4 of Figure 7-3. For Part 4R, the Controller assigned Micro_expert 4 to deal with it. According to Micro_expert 4 Rule 2, it forms an EESI model. For Part 4L, the Controller assigned Micro_expert 4 to deal with it. According to Micro_expert 4 Rule 2, it

also forms an EESI model. Thus the control returned to the Controller and the decomposition was terminated.

A number of practical considerations limit the degree of complexity which must be handled in practice:

1. there are tolerance errors for all real objects; the shape, size and relative location of the subparts will differ from the original design;
2. generally mechanical parts and other man-made objects are designed to be as simple as possible; thus many peculiar or pathological situations are unlikely to be encountered;
3. the computer vision system will introduce noise, distortion, and sampling error.

Consequently, although it would be possible to use more complex rules to exactly represent the local details at the intersection between a quadric face and several planes, we did not consider that this would be useful. Thus the resulting CSG-EESI model may be slightly different from an original BR-like model at local intersections, such as that shown in Figure 7-4. The degree of precision can be extended if that appears necessary.

The system has been implemented by using C-PROLOG under the UNIX operating system on a VAX 11/750.

To summarize, the system which has been developed has the following characteristics:

1. extensibility;

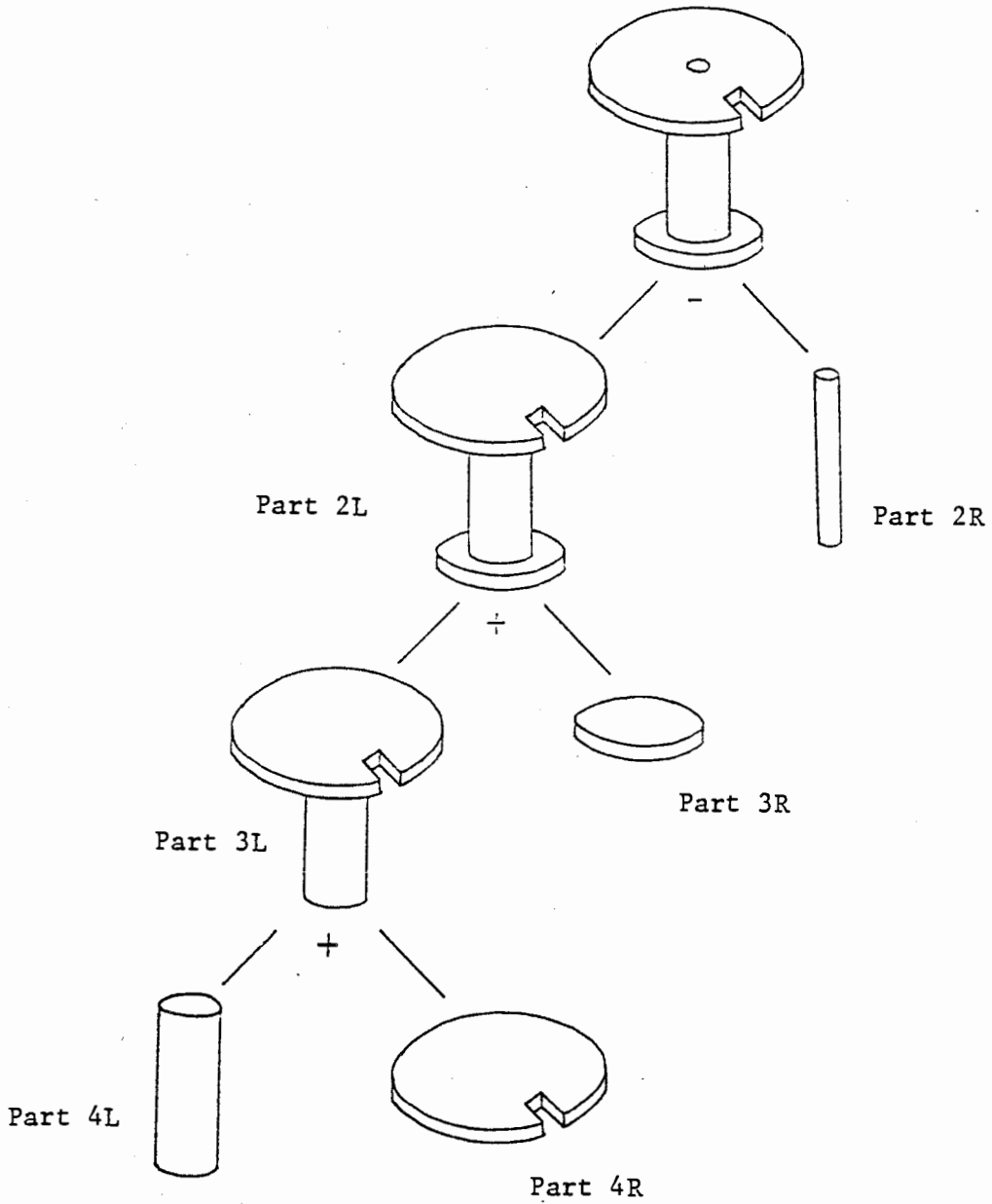


Figure 7-3: An example of a decomposition made by the BR to CSG-EESI Conversion Expert System.

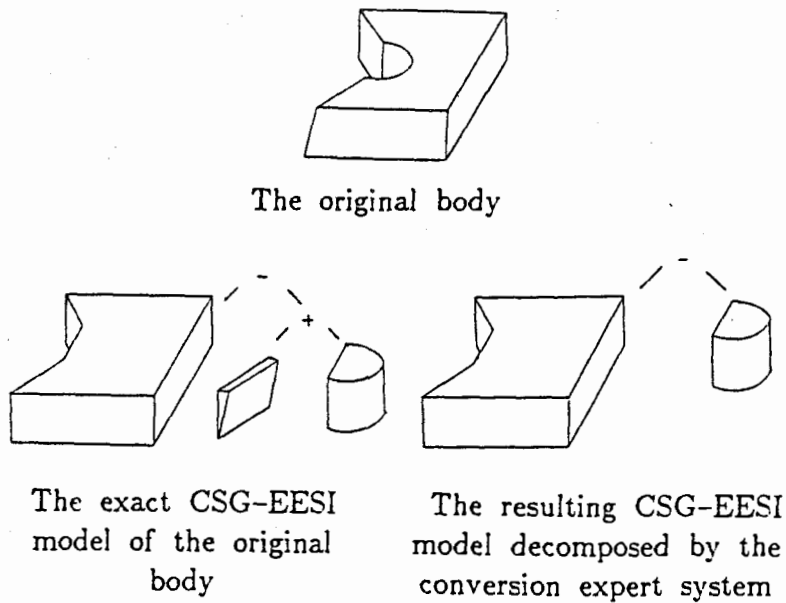


Figure 7-4: An example of the local difference which can result between an "exact" decomposition and that achieved by the expert conversion system.

2. modularity of both the knowledge base and the control mechanisms;
3. the rules in the system are based on geometric cues and thus are reliable;
4. the expert systems are well organized and built on the PROLOG language.

The system can be easily expanded to include more rules and more analysis processors in order to deal with (1) more complex intersection situations between quadric surfaces and (2) other types of quadric surfaces.

7.5. Experiments

In order to test the approach presented in this chapter, we have run two kinds of experiments with the conversion expert system. The first is to decompose bodies that have various types of faces and connectivities and the second is to decompose bodies that have similar or different kinds of structures. Figure 7-5 and 7-6 illustrate graphically the two groups of bodies which have been tested, and their decomposed CSG-EESI representations. The decomposition times using the CSG-EESI conversion expert system running under the UNIX on a VAX 11/750 are shown under each of the decomposed results in Figures 7-5 and 7-6. A typical example of the input and output format for the test of body No. 1 is shown in Appendix J.

For most bodies, the ordering of the input data will not affect the decomposition results, but in a few cases alternative decompositions may be found; this parallels human experience since there is not always a unique decomposition which is most reasonable. Since the knowledge in the conversion expert system does not relate to the geometric size of a body, other bodies which have the same structure but different size will have the same decomposition. The size differences are only reflected in their EESI models and the rigid motions in the CSG trees.

7.6. Discussion and Chapter Summary

The CSG-EESI scheme for representing bodies formed by quadric and planar faces which has been presented in this chapter offers both structural and geometrical information. Experimental results, although somewhat limited at this

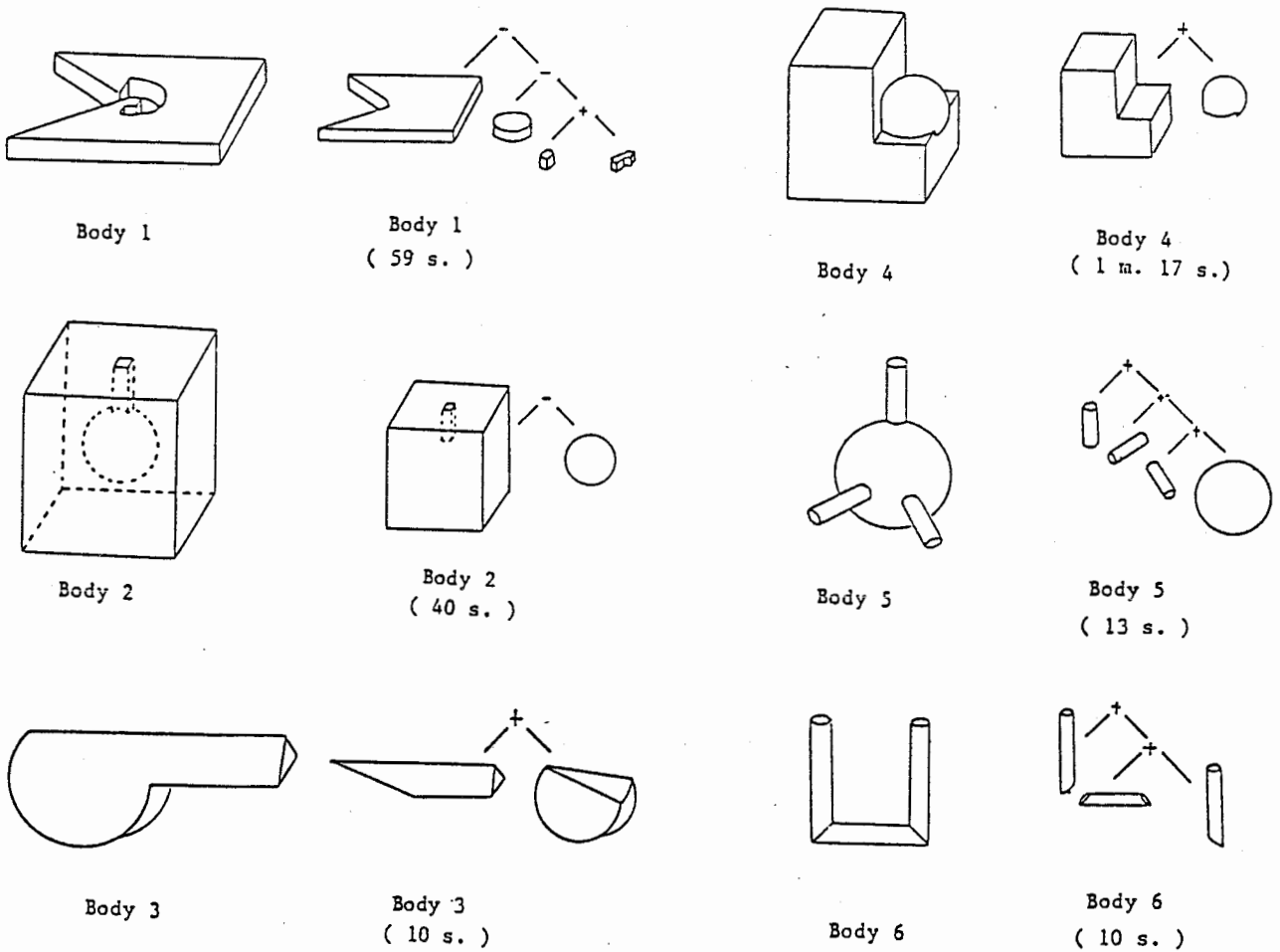


Figure 7-5: The test group with various types of faces and connectivities compares with its decomposition results. The CPU times are shown under each decomposition result.

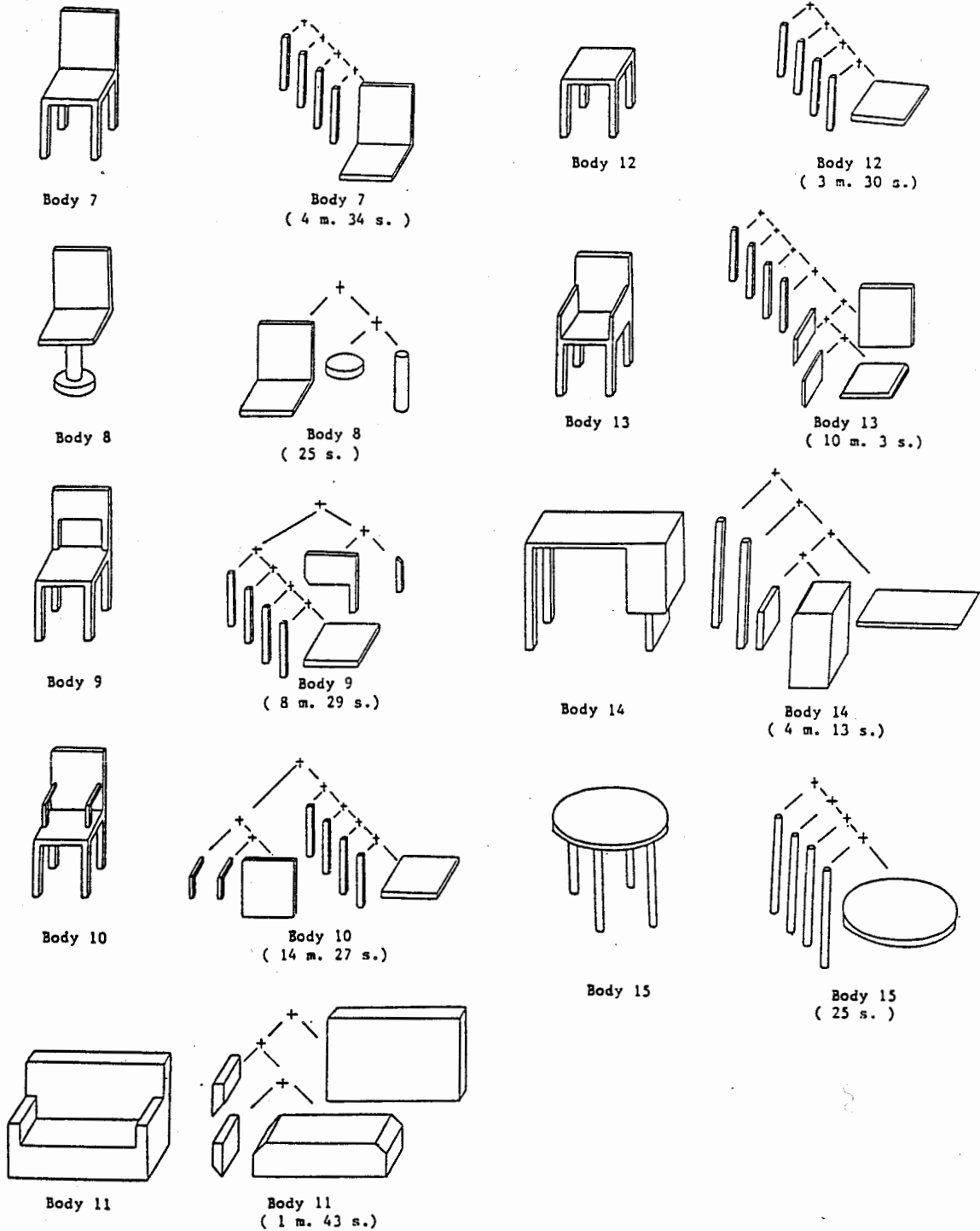


Figure 7-6: Another test group with various types of structure compares with its decomposition results. The CPU times are shown under each decomposition result.

point, show that the conversion expert system is successful and that the methodology is promising. Future expansion of the conversion expert system to include more knowledge and rules will certainly enlarge its ability to handle larger and more complex problems.

The CSG-EESI scheme may be used as the medium between pictorial models and higher level models, such as relational representations. The conversion system has been developed as part of a comprehensive robot vision system; a related part which forms BR-like models from multiple views of the bodies in a scene has been described in Chapter 3, but further work on converting CSG-EESI models to relational models or to semantic models remains to be done. This includes:

1. the approximation of primitive parts which are represented by EESI models,
2. the identification and establishment of relationships between subparts,
3. the measurement of the complexity of body structure,
4. the measurement of structural similarities between two kinds of bodies.

A standard set of model prototypes is certainly essential for object recognition. The conversion expert system which has been developed could play an important role in the automatic formation of a model database. In the proposed system illustrated in Figure 7-7, BR-like models of bodies may be constructed by means of CAD, computer vision or from 2-D model databases [70, 103, 80]; the conversion expert system (CES) then converts these BR-like models into CSG-EESI models. A higher level model could then be derived from the CSG-EESI model. Thus it is possible to automatically build a standard set of model prototypes

which are well organized and have been classified. Once a CSG-EESI model or a higher level model has been constructed, graphic techniques can be used to display the model so that model correctness could be verified by comparison with the original body.

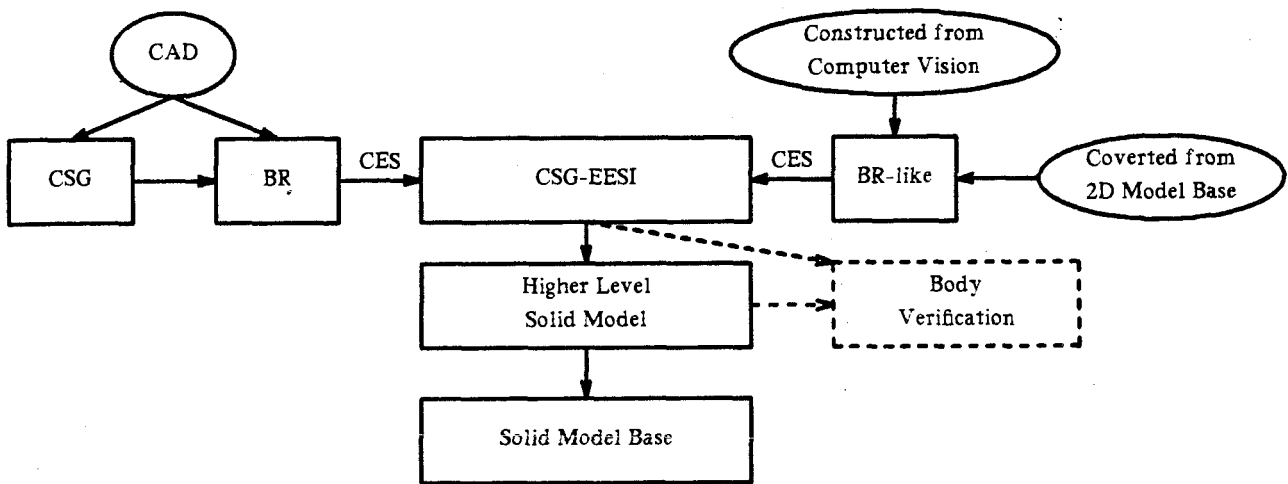


Figure 7-7: A proposed system for automatic model formation.

The CSG-EESI scheme may also be used as an internal model for the recognition of bodies when exact prototype bodies are known and built in by the same CSG-EESI conversion system. The recognition process could first construct the CSG-EESI model of the body by the means of computer vision, then reduce the matching domain based on the number of its primitive parts and the operators between primitive parts. According to the number of vectors, the number of linked vector groups, the indicators of vectors, the directions and parameters of vectors and the relationships between primitive parts, a matching primitive part

could be found in one of the candidate prototype bodies. Using the relationships between primitive parts, the matching could be propagated to the other parts. If all parts are matched, the body is recognized; otherwise, another prototype body should be tested.

In computer graphics, the conversion expert system partly fills the gap between BR and CSG representations. In the domain considered, a BR-like model can be converted into a CSG-EESI representation and each EESI primitive can be easily represented by a CSG subtree which has half-spaces and complete quadric surfaces as its primitives.

Chapter 8

The Organization and Function of the Prototype System

The purpose of this thesis is to investigate the methods needed to develop a intelligent mobile robot which is able to freely move through its environment and to recognize and manipulate the objects in the environment. In this Chapter we describe a prototype robot system which can explore a practical indoor environment and construct 3-D models of the bodies within the environment, based on sequential planned views. It should then be possible to classify bodies based on the structural and geometric information in the body models.

It was pointed out in Chapter 1 that the prototype system is a hierarchical expert system. The different levels in the hierarchical structure reflect the different control levels of the robot. The modularized system is organized in a natural way, according to its functions. Figure 8-1 shows the major modules and the organization of the prototype vision system, which consists of the following principal elements:

1. the Vision Controller which makes the decisions to control the vision process;
2. the Vision Sensor which acquires the visual data;
3. the preprocessor which deals with early and intermediate vision preprocessing;

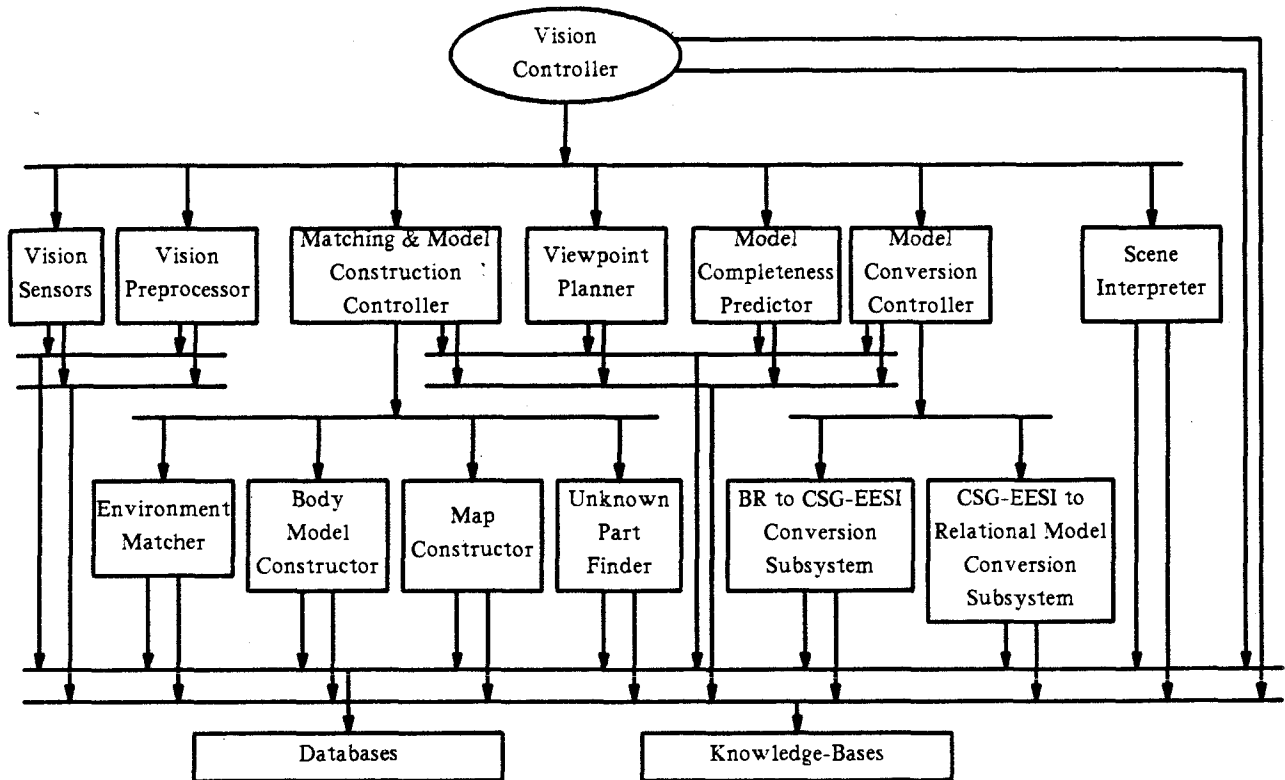


Figure 8-1: The major modules and the organization of the prototype vision system for a mobile robot.

4. the Matching and Model Construction Controller that controls the Environment Matcher which determines the self-location of the robot, the Body Model Constructor which creates, constructs and updates the body models of the scene, the Map Constructor which constructs and updates a partial map of the scene, and the Unknown Part Finder which identifies the unknown parts of the models which have been constructed so far;
5. the Viewpoint Planner which selects the next vantage view;
6. the Model Completeness Predictor which fills in the unseen parts of bodies;

7. the Model Conversion Controller that controls the process of a conversion sub-system which converts BR-like representations into CSG-EESI representations and a conversion sub-system which converts the CSG-EESI models to relational models or higher level models;
8. the Scene Interpreter which recognizes the objects in the scene;
9. the Knowledge Bases which contain the models of the environment and the general objects.

Besides the above vision system, the intelligent robot may have other sensor controllers, a planning system for its manipulator, a planning system for its movement, a controller for an inertial navigation system, and a display system which creates the reconstructed scene for the operator. All of the systems are under the control of the main controller which directs the activities of the robot. Figure 8-2 shows a schematic sketch of the control system for a mobile robot.

The vision system is an important constituent of the intelligent robot. It explores its environment, finds the available paths, determines what kind of object is in the scene, offers useful information to the manipulator to manipulate the objects, and gives valuable feedback messages to the navigation system and the manipulator for execution, monitoring and servoing. On the other hand, the navigation system offers the initial information to the vision system to determine the coordinates of the robot and, with the movement mechanism, it gives the possibility to view the environment from different positions. The manipulator may have touch sensors to confirm visual data and may move the objects around in order to let the vision system obtain a better view. The display system offers the reconstructed scene to operators, then operators can compare the reconstructed

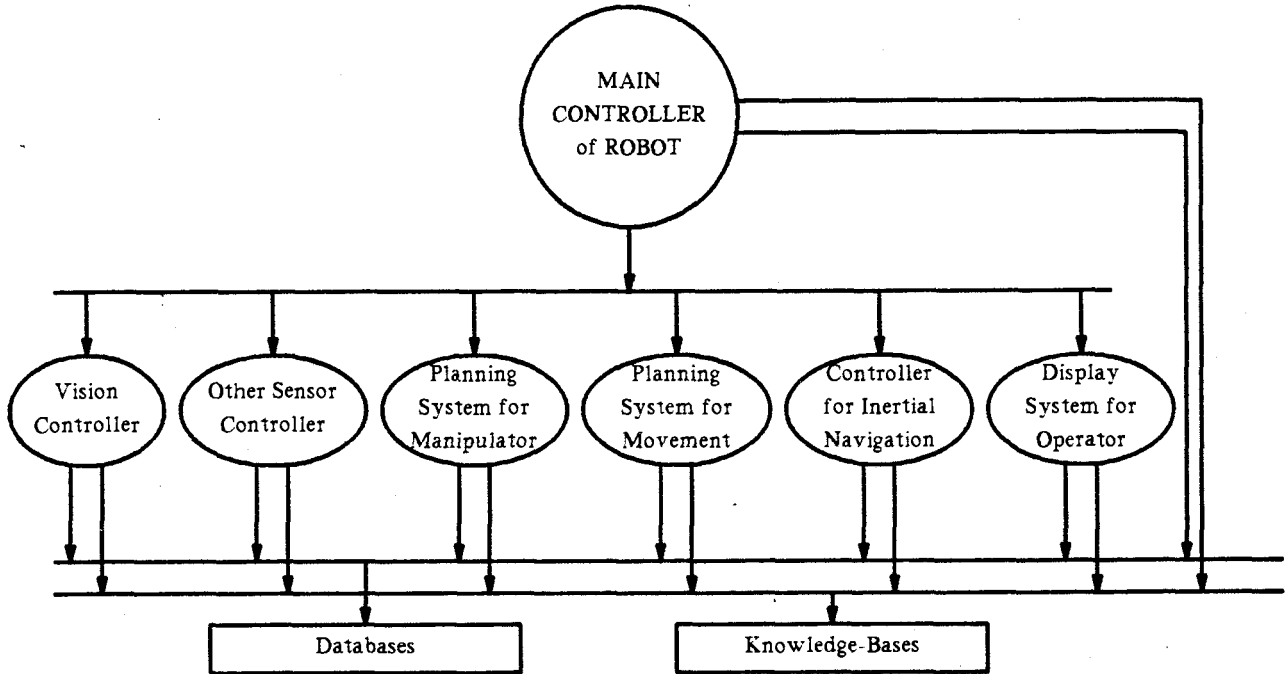


Figure 8-2: A schematic sketch of the control system for a mobile robot.

scene with television images and monitor the robot in complex situations to help the robot to make important decisions.

In the proposed vision system, framed views of the scene are taken in by the vision sensor(camera). The raw visual data are processed by a preprocessor which may use techniques, such as Stereo or Shape from Shading, to extract the internal representations as a 2 1/2 D sketch. The preprocessor provides a set of faces with their bounding lines and junctions encoded by 2-D spherical coordinates and depths measured from the viewpoint concerned. The directions of face normals are also provided. The environment model and the general object models are

stored in the knowledge bases. Those models will be used to determine the positions and navigation routes of the robot, and to recognize the bodies in the environment.

As described in Chapter 3, for each view, the partial 3-D descriptions of bodies are derived by labeling and segmenting the image. After the environment model has been matched, the partial 3-D descriptions of the first view will be used as the initially constructed partial models of the bodies in the scene. The view and the partially constructed models are checked by the Unknown Part Finder, which identifies the ambiguities. According to the types of the ambiguities, it offers the necessary information to the Viewpoint Planner. From the environment model and the partially constructed body models, the partial map is constructed by the Map Constructor which indicates the global ambiguities and the movable floor areas for the Viewpoint Planner.

Then the Viewpoint Planner will be invoked to analyze the accumulated information from previous views and select the next vantage viewpoint and view frame. After moving to that new viewpoint, the vision controller directs the sensors to take the new view in. When the new view has been accepted and preprocessed, the Matching Controller will be reinvoked to match the new view with the previously constructed models and add the new information in.

The above procedure will be repeated until the Viewpoint Planner finds that the scene has been fully explored. At that time, there may still remain some ambiguities on the bodies in the scene. Since the movement and the view area of

the robot is limited and bodies are usually put on some places or touched by some other bodies or piled together, the faces of bodies are not always visible, unless the manipulator moves or turns the bodies around. Therefore a Model Completeness Predictor usually needs to be invoked to fill in the incomplete model in order to make it complete. The Model Completeness Predictor is a sub-expert system which applies general assumptions, such as smoothness and extensibility of faces and edges, and other heuristic rules to the visual data.

The body models (either partial or complete) which are constructed from the multiple views have representations that are similar to a Boundary Representation (BR). Once a complete model has been constructed, the Model Conversion Controller will convert the model into higher level representations. One conversion system described in Chapter 7 is used to transform the BR representations into our new CSG-EESI representations. The CSG-EESI representation provides both structural and geometric information for bodies and facilitates object classification by allowing comparison of unknown object structures with those prototype objects which might be expected to be in the environment. The CSG-EESI representation offers a good foundation for deriving higher level 3-D models, such as relational models.

In the system knowledge bases, the prototype objects are organized in suitable formats which will allow the interpreter to search them and compare them with an unknown body. The final result of analyzing a scene by the system is an aerial map of the scene, the 3-D models for the bodies within the environment and their corresponding names. The map indicates the locations of the bodies and the

relationships between them. Also the trajectory of the robot movement is recorded and displayed on the map. The map will be stored and used to plan the other activities of the robot. For patrol robots, methods have been developed in Chapter 5 to plan optional routes through environments for which the map is complete.

The prototype vision system shown in Figure 8-1 is certainly complex. In this thesis, we have investigated some important components, such as the Matching and Model Construction Controller and its subsystems, the Viewpoint Planner and the BR to CSG-EESI Conversion Subsystem, but other components remain which need further exploration.

Chapter 9

Conclusions

In this thesis, new methodologies and techniques have been investigated and developed for a vision system of a mobile intelligent robot. These include:

1. a new matching and model construction technique which matches a view with the environment model, matches widely separated views and sequentially constructs the 3-D body models,
2. a new viewpoint planning method which plans the vantage views based on the previously accumulated information of the bodies and their relationships with the environment,
3. a new 3-D representation scheme (CSG-EESI representation) and its conversion expert system which converts BR-like representations into the CSG-EESI representation.

A prototype system, which has been partially implemented, has also been described in this thesis. This vision system explores the environment and constructs 3-D models of the bodies in an indoor scene by automatically selecting vantage views.

The prototype vision system is an elegant combination which synthesizes different kinds of cues (e.g., photometric, topologic, geometric constraints and wide angle stereo), vision with planning, numerical methods with symbolic analysis, different internal representations, and different expert systems (they consist of different types of knowledge and different reasoning schemes).

The characteristic features of the vision system are as follows:

1. The rules in the system are based on physical, topologic and geometric constraints. Thus they are reliable.
2. The redundant information gathered from vision is fully utilized to help forward reasoning. Thus the search space is greatly reduced.
3. Spherical projection images are used for analysis of a scene with a wide angle of view and various depths of field.
4. A well organized vision expert system is built using the PROLOG language.

In practice, a mobile robot should be able to perform its functions as fast as necessary, thus real time visual processing is required. In Figure 1-2, if each block of sub-expert and micro-expert system were replaced by a PROLOG micro-computer, then the hierarchical structure of the expert system would offer a good model for such a network of micro-computers which could effectively deal with complex vision tasks. Still further, such a network can lay a foundation for parallel and pipeline vision analysis.

Open Problems

Automatic reconstruction and recognition of a complex scene requires the use of many sources of knowledge and involves many research topics. The following are only a few of the open problems related to our prototype vision system:

1. We have restricted our research to bodies for which the vertices are formed by at most three surfaces and the edges are formed by two surfaces. To include a wider range of body types is a natural extension.

2. Intelligent robots have not only a brain and a vision system, but also a pair of hands. To co-ordinate the prototype vision system with a manipulator system would be of great value to industry. In certain situations, such a co-ordinated system would greatly simplify and speed up the scene analysis process, e.g., a stacking robot picks out a body from a bin, then turns it around and recognizes it. In many cases, some important relationship concepts, such as "touched by" and "connected by", are not easy to be distinguished by only the means of vision. But, under the co-ordinated system, the distinctions could be easily found (e.g., pull objects).
3. In this prototype vision system, the image analysis under consideration begins with perfect 2 1/2 D sketches and produces 3-D CSG-EESI models of bodies. At this time, there is no single algorithm or scheme that has given a satisfactory method to extract 2 1/2 D sketches from real images, but some ideas seem to be attractive approaches. One idea is to combine different cues to form an elegant analysis system for solving the problem. Another idea is to combine the bottom-up and top-down strategies for solving the ambiguities of the low level image processing. To devise such a low level image processing system and to co-ordinate it with the prototype vision system would result in significant research.
4. The characteristics which can be used to classify objects include: geometric properties, physical properties, frequency of occurrence, usage and functional characters. Besides the geometric properties, colour and texture are two significant and important cues for human vision. Thus, extending the system to include more cues is one of the big open problems.
5. Much further work on converting CSG-EESI models to relational models or to semantic models remains. This includes: the approximation of primitive parts which are represented by EESI models, the recognition and the establishment of relationships between subparts, the measurement of the complexity of body structure, the measurement of structural similarities between two bodies, and the classification of bodies.
6. The use of a hardware network to analogize the expert system structure would be a meaningful extension. Based on such a network, the parallel and pipeline processing of sequential images would be an interesting research topic.

7. The implementation of a mobile robot with a vision system of the type described would lay a solid foundation for all of the above mentioned research work.

Appendix A

Format of Input and Output Data

We assume that all of the early and some of the intermediary visual computations are done by a kind of pre-processor. Thus we start with the descriptions that could be obtained easily from a perfect 2 1/2 D sketch.

A perfect 2 1/2 D sketch means that surface orientations, depths and discontinuity information are available for the whole viewport. And the quantities of surface orientations and depths are available with high precision.

The information which can be easily extracted from a perfect 2 1/2 D sketch and the image intensity includes:

1. Category of Facet, i.e., planar, conical, cylindrical or spherical.
2. Type of Region & Facet, i.e., light (l), light-shadow (l_s) or shadow (s).
3. Category of Line & Edge, i.e., straight line, circle, other curve.
4. Type of Line & Edge, i.e., shadow, occluding boundary, concave, convex, clipping line, limb.
5. The Coordinates of Junction & Vertex, i.e., spherical coordinates, depth from a viewpoint.
6. Category of Junction, i.e., vertex, virtual junction, clipping point, shadow intersection point (SIP), frame point.

7. Type of Junction, i.e., w, t, p, s, a, q, m, y, x, k... [Reference to the definitions given in Appendix C]

8. Orientation of Planes.

Since the 2 1/2 D sketch is provided, it is reasonable to assume that the above information will be offered by the pre-processor.

The components and format of the input to the matching subsystems is as follows:

1. The format of the model of the surrounding environment:

- a. environment(type, place_list, frontier_list);
- b. place(name, frontiers, surface_number, vertex_number, dimensions, surface_list);
- c. frontier(name, composing_places, located_surfaces, symmetric_center_location, shape_parameters, part_faces);
- d. face(name, category, normal/axis, distance_from_center, convex_connected_faces, concave_connected_faces, part_edges, center_coord, Part/full);
- e. edge(name, category, type, part_points, composing_faces, length, parameters);
- f. point(name, category, type, composing_lines, composing_faces, __, __, 3D_coord, error_3D, center_coord);

2. frame(name, viewpoint_coord¹⁶, predict_viewpoint, error_viewpoint*, reference_point_coord, viewline, twist, view_size, local/global, region_list, line_list, junction_list);

^{16*} indicates that the variable may have a null value.

3. region(name, category, type, part_lines, area*, body_name*, touch*, occluded_by*, plane_normal);
4. line(name, category, type*, part_junc, compose_faces*, length**17, parameters, part/full*);
5. junction(name, category, type*, composing_lines, composing_faces*, sph_coord, depth**, 3D_coord*, error_sph*); (Note : sph_coord = [U, V])

The input data can be viewed as relational databases with null values.

The format of the output for the matching and model construction processes is as follows:

1. scene(body_list, part/full);
2. body(name, planes, convex_quadric_faces, concave_quadric_faces, touch/by, boundary_list, part/full, face_list, edge_list, point_list);
3. face(name, category, normal/axis, distance_from_center/parameters, convex_connected_faces, concave_connected_faces, body_name, part_edges, center_coord, point_on_it, touch/by, part/full);
4. edge(name, category, type, part_points, composing_faces, length, parameters, part/full);
5. point(name, category, type, composing_lines, composing_faces, sph_coord, depth, 3D_coord, error_3D, center_coord); (Note : sph_coord = [Frame, [U, V]])
6. the lists of matched faces, edges or points for different views.

The input format for the CSG-EESI conversion step is as follows:

^{17**} indicates that the variable may have a value with a tolerance error.

1. body(name, planes, convex_quadric_faces, concave_quadric_faces).
2. face(name, category, normal/axis, distance_from_center/parameters, convex_connected_faces, concave_connected_faces, center_coord, point_on_it);
3. point(name, composing_faces, 3D_coord);

The output format for the CSG-EESI conversion step is as follows:

1. body(name, complexity*, small_box*, deficiency*, number_of_surface, number_of_vertices, CSG_EESI_R);
2. subpart(name, complexity*, small_box*, deficiency*, number_of_surface, number_of_vertices, CSG_EESI_R);
3. The format of the EESI representation (EESI_R) is [vector_list, [uni, void, void]].
4. The format of the CSG-EESI representation(CSG_EESI_R) is as [face_list, [operator, CSG_EESI_R, CSG-EESI_R]] or [face_list, [operator, EESI_R, EESI_R]].
5. vector(name, category_indicator, normal/axis_direction, distance_from_center/parameters, concave_connected_planes/linked_list, center_coord, _).

The internal databases of the system have the following formats :

1. view_frame_stack :

frame(name, viewpoint_coord, reference_point_coord, twist, view_size, local/global, region_list, line_list, junction_list);

2. body models:

body(name, planes, convex_quadric_faces, concave_quadric_faces, touch/by, boundary, part/full);

face(name, category, normal/axis, distance_from_center/parameters, convex_connected_faces, concave_connected_faces, body_name, part_edges, center_coord, point_on_it, touch/by, part/full);

edge(name, category, type, part_points, composing_faces, length, parameters, part/full);

point(name, category, type, composing_lines, composing_faces, 3D_coord, error_3D, center_coord);

The format of the constructed map is as follows:

1. map(name, environment_name, body_list, trajectory_list).
2. environment(name, projected_line_list, projected_point_list).
3. body(name, projected_line_list, projected_point_list).
4. projected_line(name, category, end_points_list, body_name).
[Note: Here category is Projected Viewline (PVL), Confirmed Boundary Edge (CBE) or Approximated Boundary Edge (ABE).]
5. projected_point(name, category, coord).
[Note: Here category is Projection of Vertex, Projection of "t" Type Point or Projection of Other Feature Point.]
6. trajectory(name, viewpoint_list, time_interval).
7. viewpoint(name, time, frame_name).

Appendix B

Rules for Matching Multiple Views

We use the multi-level matching method to establish the correspondence between two views taken from planned viewpoints.

Matching is held in the following three different levels: point level, facet level, and body level. In each level, the positions and the characteristics of the features in this level, and the relationships between features on the same level or between features on adjacent levels may be used as cues for matching.

For points, those cues include: coordinates, junction categories and types, connected groups of points, and the information from related matched edges, facets or bodies.

In the facet level,

1. for an edge, those cues include: its category, type, direction, length and convexity, the information from related vertices, its relationships with other edges, and the information from the related matched facets or bodies;
2. for a facet, those cues include: its category, direction, area, convexity, and number of edges and vertices, the information from related vertices and edges, its relationships with other facets, and the information from the related matched bodies.

For bodies, those cues include: its position, dimensions, shape, structure, and consisted facets, the information from related vertices, edges and facets, and the relationships with other bodies.

Rules for matching different views are set out as follows:

1. The rules derived from the limitations of the research domain:

- a. Since the bodies are solid,
 - i. the category of a facet, the type of an edge, the convexity of an edge or a facet, the relationships between facets and the topological characteristics of the bodies are unchangeable in different views;
 - ii. the types of those edges and surfaces which form a vertex do not change in different views;
 - iii. the types and convexities of surfaces which form an edge do not change in different views.
- b. Since the scene is static, the bodies in it do not actively change their locations. Therefore the 3D coordinates of a vertex is unchangeable between views.
- c. Since the scene is static and the bodies are solid, the directions of those planes which form a vertex do not change between views and the directions of those planes which form an edge do not change also.
- d. If the light source is fixed, then the shadows in a static scene are fixed too. The locations of those feature points which relate to shadows are also fixed.

2. The rules derived from uniqueness:

- a. Each feature point in one view can only match zero or one feature point in another view.

- b. Except new occlusions occur, the projection of a feature point will not disappear in the new view.

3. The rules derived from consistence of features:

- a. In general, images taken from different viewpoints will appear different. A feature point will change its project in a new spherical coordinate system that corresponds to a new viewpoint, except the feature point exactly locates on the line that passes through the two viewpoints. But the two projection points that correspond to a same feature point in two views and the two origins of the spherical coordinate systems that correspond to the two viewpoints should locate at a same plane.
- b. When a viewpoint changes, the junction types that correspond to a same vertex can only transit in its own family as Appendix E shows.
- c. The junction types in a view should be consistent, therefore a local relaxation method may be used to determine the junction label changements.

4. The rules derived from triangulation:

- a. If two projections of a feature point have been matched and the locations of the two origins of spherical coordinate systems that correspond to the two viewpoints are known, then the 3-D orthogonal coordinates of the feature point can be calculated.
- b. If the 3-D orthogonal coordinates of a feature point are known, then the spherical coordinates of its projection in different views can be calculated.

5. The rules derived from photometrics:

- a. If an edge is a straight line segment, then its projection should be a straight line segment or a point.
- b. If an edge is a circle, then its projection should be a straight line segment, a circle or an ellipsis.

6. The relationships between a known feature and its related features can be used as cues to match the related features in different views. In this case, the known feature is called Reference Feature.
 - a. If a body touches a Reference Body, the touch relationship is a cue for matching the projections of the body.
 - b. If a shadow of a Reference Body casts on another body, then the shadow can be used as a cue for matching the projections of the another body.

7. The rules derived from unchangeable topological characteristics:
 - a. If a vertex matches another vertex in another view, then an edge incident from the vertex should matches another edge incident from the another vertex except the another edge is occluded.
 - b. If a vertex on an edge (a facet or a body) and the edge (facet or body) matches another edge (facet or body) in another view, then the matched vertex in the another view should locate on the another edge (facet or body). Especially, if the vertex is the intersection point of edges (facets), the matched vertex in the another view is also the intersection point of the matched edges (facets).
 - c. If an edge on a facet (a body) and the facet (body) matches another facet (body) in another view, then the matched edge in the another view should locate on the another facet (body). Especially, if the edge is the intersection edge of facets, the matched edge in the another view is also the intersection edge of the matched facets.
 - d. If a facet on a body and the body matches another body in another view, then the matched facet in the another view should locate on the another body.
 - e. For related features in a same level, if one matches a feature in another view and its relationships are reserved, then its related features should match with those correspondents in the other view except the correspondents are not exist.

- i. If a facet and one of its edges are matched, then its adjacent facet which is adjacent by the matched edge should match with the corresponding facet in the other view except the correspondent does not exist.
- ii. If a facet, one of its edges and one vertex of the edge are matched; then the edge which is adjacent by the vertex and on the same facet should match the corresponding one in the other view except the correspondent does not exist.

8. The rules derived from the laws of geometry:







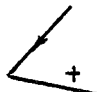
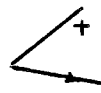

- a. If two vertices in one view match other two in another view, then the straight line segment ended by the two vertices should match the straight line segment ended by the other two.
- b. If three vertices in one view match other three in another view, then the plane passing through the three vertices should match the plane passing through the other three.
- c. If three vertices in one view match other three in another view, then the circle passing through the three vertices should match the circle passing through the other three.
- d. If a vertex in one view matches another one in another view and the two planes, on which the two points are located respectively, have the same normal direction; then the two planes are matched.
- e. If a quadric surface of a body in one view matches another surface of another body in another view, then the two bodies are matched.
- f. If a plane of a body and a point, which is on the body but not on the plane, match another plane and another point of another body in another view; then the two bodies are matched.
- g. If four points of a body which do not locate on a same plane match other four points of another body in another view, then the two bodies are matched.
- h. If two facets of a body in one view match the other two facets

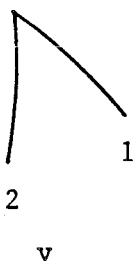
of another body in another view, then the two bodies are matched.

9. The rules derived from the laws of physics on relative movement:
 - a. If a robot closes in on (moves away from) a body, then the projection of the body will enlarge (reduce) its size.
 - b. If the distance between a body and a viewpoint is closer (farther) than the distance between a reference body and the viewpoint, then if the viewpoint moves left, the projection of the body will move right (left) relatively to the projection of the reference body; if the viewpoint moves right, up or down, similar results can be drawn.
 - c. Occlusions offer cues of the distances between bodies and a viewpoint.
10. When the robot only stretches its neck, then $\phi_n = \phi_o$, $\theta_n < \theta_o$, if the corresponding projections have spherical coordinates (ϕ_o, θ_o) and (ϕ_n, θ_n) for the old and new views respectively.

Appendix C

Junction Labeling Scheme

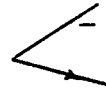
Junction type	Junction Label	Label of Each Line		Waltz' Label	
	0v	0	0		
	1v1	1	1		
	1v2	1b	1		
	1v3	1	1b		
	1v4	1m	1m		
	1v5	1m	1		
	1v6	1	1m		
	2v1	1	2a		
	2v2	2a	1		
	2v3	1	2b		



Junction type	Junction Label	Label of Each Line	Waltz' Label
---------------	----------------	--------------------	--------------

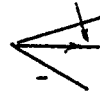
2v4

2b 1



1w1

0 1 1b



1w2

0 1b 1



1w3

1b 1 0



1w4

1 1b 0



1w5

1m 1m 0



1w6

0 1m 1m



2w1

1 2a 1



2w2

1b 2a 1b



2w3

1 0 2a



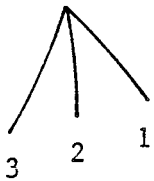
2w4

2a 0 1



2w5

1m 2a 1m

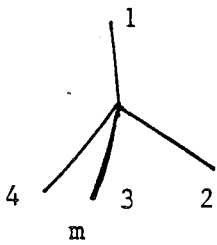


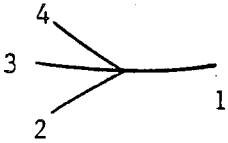
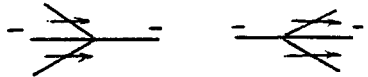
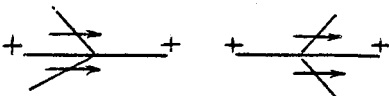
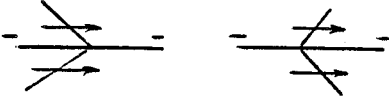
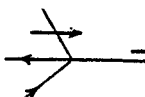
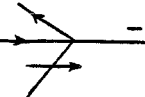
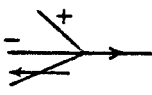
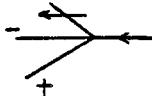
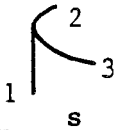


w

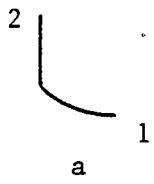
Junction type	Junction Label	Label of Each Line			Waltz' Label
	3w1	2a	2b	2a	
	3w2	2b	2a	2b	
	1y1	1b	0	1	
	1y2	0	1b	1	
	2y1	1	2b	1	
	2y2	1b	2a	1b	
	2y3	1b	2b	1b	
	2y4	1m	2b	1m	
y	3y1	2a	2a	2a	
	3y2	2b	2b	2b	
	3y3	2b	2a	2b	
	1t1	1	0	1	

Junction type	Junction Label	Label of Each Line			Waltz' Label	
	1t2	1m	0	1m		
	2t	2a	0	2a		
	11t1	1	1	1		
	11t2	1	1b	1		
	11t3	1m	1	1m		
	11t4	1m	1b	1m		
	11t5	1m	1m	1m		
	11t6	1b	1	1		
	11t7	1	1	1b		
	11t8	1b	1m	1		
	11t9	1	1m	1b		
	11t10	1	1m	1		
	21t1	1	2a	1		

Junction type	Junction Label	Label of Each Line				Waltz' Label
	21t2	1	2b	1		
	21t3	1m	2a	1m		
	21t4	1m	2b	1m		
	21t5	1b	2a	1		
	21t6	1	2a	1b		
	2p1	1b	2a	1b	0	
	2p2	0	1b	2a	1b	
	3p1	2a	2b	0	2a	
	3p2	2a	0	2b	2a	
	21k1	1	0	2a	1b	
	21k2	1b	2a	0	1	

Junction type	Junction Label	Label of Each Line				Waltz' Label	
	2m1	1	1	1b	0		
	2m2	1	1	0	2b		
	1x	1b	0	1b	0		
	2x1	2a	0	2a	0		
	2x2	2b	0	2b	0		
	11x1	1b	1	1	0		
	11x2	1b	0	1	1		
	12x1	1	0	1b	2a		
	12x2	1	2a	1b	0		
	2s	1m	1	2a			

Junction type	Junction Label	Label of Each Line	Waltz' Label
------------------	-------------------	-----------------------	--------------

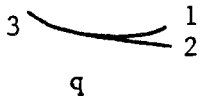


1a1

1b 1m

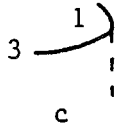
1a2

1 1m



1q

1b 0 1b



2c

2a 1

Appendix D

Knowledge for Labeling Junctions and Their Related Lines

The following is the rules for labeling junctions and related lines. The rules are stored separately in micro-knowledge-bases according to the categories of junctions. Each rule consists of 3 parts: known facts of the related lines, new labels of the related lines, and the label of the junction.

1. Rules which classify the 'p' type junctions:

- a. rule([_, _, _, 0], ['1b', '2a', '1b', _], '2p1').
- b. rule([0, _, _, _], [_, '1b', '2a', '1b'], '2p2').
- c. rule([_, _, 0, _], ['2a', '2b', _, '2a'], '3p1').
- d. rule([_, 0, _, _], ['2a', _, '2b', '2a'], '3p2').

2. Rules which classify the 'w' type junctions:

- a. rule([0, _, b], [_, '1', '1b'], '1w1').
- b. rule([0, b, _], [_, '1b', '1'], '1w2').
- c. rule([b, _, 0], ['1b', '1', _], '1w3').
- d. rule([_, b, 0], ['1', '1b', _], '1w4').
- e. rule(['1m', '1m', 0], [_, _, _], '1w5').

- f. rule([0, '1m', '1m'], [_, _, _], '1w6').
- g. rule([b, a, b], ['1b', '2a', '1b'], '2w2').
- h. rule(['1', a, _], [_, '2a', '1'], '2w1').
- i. rule([_, a, '1'], ['1', '2a', _], '2w1').
- j. rule(['1', _, '1'], [_, '2a', _], '2w1').
- k. rule([a, _, a], ['2a', '2b', '2a'], '3w1').
- l. rule([_, 0, a], ['1', _, '2a'], '2w3').
- m. rule([a, 0, _], ['2a', _, '1'], '2w4').
- n. rule(['1', a, _], [_, '2a', '1'], '2w1').
- o. rule([_, a, '1'], ['1', '2a', _], '2w1').
- p. rule(['1m', a, '1m'], [_, '2a', _], '2w5').
- q. rule(['2b', '2a', _], [_, _, '2b'], '3w2').
- r. rule([_, '2a', '2b'], ['2b', _, _], '3w2').

3. Rules which classify the 'y' type junctions:

- a. rule([a, a, _], ['2a', '2a', '2a'], '3y1').
- b. rule([_, a, a], ['2a', '2a', '2a'], '3y1').
- c. rule([a, _, a], ['2a', '2a', '2a'], '3y1').
- d. rule([b, 0, _], ['1b', _, 1], '1y1').
- e. rule([_, b, 0], [1, '1b', _], '1y1').
- f. rule([0, _, b], [_, 1, '1b'], '1y1').
- g. rule([0, b, _], [_, '1b', 1], '1y2').

- h. rule([_, 0, b], [1, _, '1b'], '1y2').
- i. rule([b, _, 0], ['1b', 1, _], '1y2').
- j. rule(['2b', '2b', b], [_, _, '2b'], '3y2').
- k. rule([b, '2b', '2b'], ['2b', _, _], '3y2').
- l. rule(['2b', b, '2b'], [_, '2b', _], '3y2').
- m. rule([a, '2b', b], ['2a', _, '2b'], '3y3').
- n. rule([a, b, '2b'], ['2a', '2b', _], '3y3').
- o. rule([b, a, '2b'], ['2b', '2a', _], '3y3').
- p. rule(['2b', a, b], [_, '2a', '2b'], '3y3').
- q. rule(['1b', '1b', b], [_, _, '2b'], '2y3').
- r. rule([b, '1b', '1b'], ['2b', _, _], '2y3').
- s. rule(['1b', b, '1b'], [_, '2b', _], '2y3').
- t. rule(['1', '1', b], [_, _, '2b'], '2y1').
- u. rule([b, '1', '1'], ['2b', _, _], '2y1').
- v. rule(['1', b, '1'], [_, '2b', _], '2y1').
- w. rule(['1b', '1b', a], [_, _, '2a'], '2y2').
- x. rule([a, '1b', '1b'], ['2a', _, _], '2y2').
- y. rule(['1b', a, '1b'], [_, '2a', _], '2y2').
- z. rule(['1m', b, '1m'], [_, '2b', _], '2y4').
- aa. rule(['1m', '1m', b], [_, _, '2b'], '2y4').
- bb. rule([b, '1m', '1m'], ['2b', _, _], '2y4').

4. Rules which classify the 'v' type junctions:

- a. $\text{rule}([0, _], [_, 0], '0v')$.
- b. $\text{rule}([_, 0], [0, _], '0v')$.
- c. $\text{rule}(['1', '1'], [_, _], '1v1')$.
- d. $\text{rule}([b, _], ['1b', '1'], '1v2')$.
- e. $\text{rule}([_, b], ['1', '1b'], '1v3')$.
- f. $\text{rule}(['1m', '1m'], [_, _], '1v4')$.
- g. $\text{rule}(['1m', '1'], [_, _], '1v5')$.
- h. $\text{rule}(['1', '1m'], [_, _], '1v6')$.
- i. $\text{rule}([_, a], ['1', '2a'], '2v1')$.
- j. $\text{rule}([a, _], ['2a', '1'], '2v2')$.
- k. $\text{rule}([_, '2b'], ['1', _], '2v3')$.
- l. $\text{rule}(['2b', _], [_, '1'], '2v4')$.

5. Rules which classify the 'a' type junctions:

- a. $\text{rule}([b, '1m'], ['1b', _], '1a1')$.
- b. $\text{rule}(['1', '1m'], [_, _], '1a2')$.

6. Rules which classify the 'x' type junctions:

- a. $\text{rule}([b, 0, _, 0], ['1b', _, '1b', _], '1x')$.
- b. $\text{rule}([_, 0, b, 0], ['1b', _, '1b', _], '1x')$.
- c. $\text{rule}([a, 0, _, 0], ['2a', _, '2a', _], '2x1')$.

- d. $\text{rule}([_ , 0, a, 0], ['2a', _ , '2a', _], '2x1')$.
- e. $\text{rule}(['2b', '0', '2b', '0'], [_ , _ , _ , _], '2x2')$.
- f. $\text{rule}([b, _ , _ , 0], ['1b', 1, 1, _], '11x1')$.
- g. $\text{rule}([b, 0, _ , _], ['1b', _ , 1, 1], '11x2')$.
- h. $\text{rule}([_ , 0, b, a], [1, _ , '1b', '2a'], '12x1')$.
- i. $\text{rule}([_ , a, b, 0], [1, '2a', '1b', _], '12x2')$.

7. Rules which classify the 'q' type junctions:

- a. $\text{rule}([_ , 0, _], ['1b', _ , '1b'], '1q')$.

8. Rules which classify the 'm' type junctions:

- a. $\text{rule}([_ , _ , b, 0], [1, 1, '2b', _], '2m1')$.
- b. $\text{rule}([_ , _ , 0, b], [1, 1, _ , '2b'], '2m2')$.

9. Rules which classify the 's' type junctions:

- a. $\text{rule}(['1m', _ , a], [_ , 1, '2a'], '2s')$.

10. Rules which classify the 'c' type junctions:

- a. $\text{rule}(['2a', _ , 1], [_ , _ , _], '2c')$.

11. Rules which classify the 't' type junctions:

- a. $\text{rule}(['1', 0, _], [_ , _ , '1'], '1t1')$.
- b. $\text{rule}([_ , 0, '1'], ['1', _ , _], '1t1')$.
- c. $\text{rule}(['1m', 0, '1m'], [_ , _ , _], '1t2')$.
- d. $\text{rule}([a, 0, a], ['2a', _ , '2a'], '2t')$.

- e. rule(['1', '1', '1'], [_, _, _], '11t1').
- f. rule(['1', '1b', '1'], [_, _, _], '11t2').
- g. rule(['1m', '1', '1m'], [_, _, _], '11t3').
- h. rule(['1m', '1b', '1m'], [_, _, _], '11t4').
- i. rule(['1m', '1m', '1m'], [_, _, _], '11t5').
- j. rule(['1b', '1', '1'], [_, _, _], '11t6').
- k. rule(['1', '1', '1b'], [_, _, _], '11t7').
- l. rule(['1b', '1m', '1'], [_, _, _], '11t8').
- m. rule(['1', '1m', '1b'], [_, _, _], '11t9').
- n. rule(['1', '1m', '1'], [_, _, _], '11t10').
- o. rule(['1', 'a', '1'], [_, '2a', _], '21t1').
- p. rule(['1', '2b', '1'], [_, _, _], '21t2').
- q. rule(['1m', 'a', '1m'], [_, '2a', _], '21t3').
- r. rule(['1m', 'b', '1m'], [_, '2b', _], '21t4').
- s. rule([b, a, _], ['1b', '2a', '1'], '21t5').
- t. rule([_, a, b], ['1', '2a', '1b'], '21t6').

12. Rules which classify the 'k' type junctions:

- a. rule(['1', 0, a, b], [_, _, '2a', '1b'], '21k1').
- b. rule([b, a, 0, '1'], ['1b', '2a', _, _], '21k2').

Appendix E

The Dictionary of Junction Families

The following is the Dictionary of Junction Families. In it, each list is a family of junction which consists of the possible junction types for a specific kind of vertex when the vertex is viewed from different positions.

['0v'].

['1v1', '2w1', '3y1', '21t1'].

['1v1', '2v1', '2v2', '3w2', '2y1', '21t2'].

['1v1', '2v1', '2v2', '2w3', '3w1', '3p1', '2m1'].

['1v1', '2v1', '2v2', '2w4', '3p2', '2m2'].

['1v2', '1w2', '2p2', '1y2', '21t3'].

['1v2', '1v3', '1w3', '2w2', '21t3'].

['1v3', '1w1', '2p2', '1y1', '21t3'].

['1v4', '1w5', '1w6'].

['1v5', '1v6', '2w5'].

['1q'].

['1a1'].

['2s', '1a2'].

['1x', '1t1'].

['2x1', '1t1'].

Appendix F

The Mathematical Description of a Quadric Surface or a Curve

In general the mathematical equation of a quadric surface has nine coefficients. These coefficients can be determined by suitably selecting enough points on the surface. An exact method of determining the coefficients from known data points involves directly solving a set of nine independent linear equations. This means that at least nine independent data points are required here for the solution. If a larger number of data points is available, then a least mean square error solution can be obtained. In some special situations, fewer data points are required:

1. If a surface is known to be spherical, then the coefficients of its equation can be determined by four data points on the surface.
2. If the direction of the axis of symmetry is known, then the coefficients of the equation of a circular cylindrical or conical surface can be determined by six data points on the surface (Note: its general equation is $aX^2+bY^2+cZ^2+dX+eY+fZ-1=0$).
3. If the direction of the axis of symmetry of a circular cylindrical surface is known, then the coefficients of the surface equation can be determined by four data points on the cylindrical surface (Note: its general equation is $a(X^2+Y^2)+cX+dY+eZ-1=0$).

In Appendix G, the methods and mathematical formulas of calculating the center and radius of a spherical surface and the axis of a cylindrical or conical surface from two views have been given.

A space curve is used to be described by a pair of surface equations, such as

$$\begin{cases} F(X, Y, Z) = 0 \\ G(X, Y, Z) = 0. \end{cases}$$

where F and G are quadratic equations. If F and G are determined, the space curve is also determined.

For a planar quadric curve, three known points on the plane determines a transformation which transforms the plane to the X - Y plane of the fixed orthogonal coordinate system. Thus, the general equation of a planar quadric curve could be as

$$aX^2 + bXY + cY^2 + dX + eY - 1 = 0.$$

If five more data points on the planar quadric curve are known, then the coefficients of the equation can be determined.

Appendix G

Some Mathematical Formulas of Spherical Projection

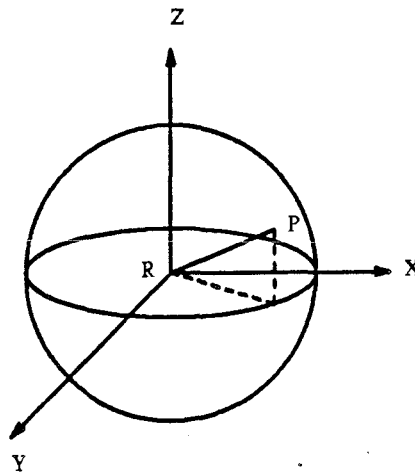


Figure G-1: The spherical coordinate system based on the robot's eye.

1. Figure G-1 shows a spherical coordinate system based on the robot's eye. R is the origin, the radius of the sphere is 1 unit and an image pixel P is represented by the pair (ϕ_p, θ_p) .
2. The conversion of the coordinates between the fixed orthogonal coordinate system and a spherical coordinate system is as the follows:

$$\begin{cases} X_p = X_r + \cos\phi_p \sin\theta_p \\ Y_p = Y_r + \sin\phi_p \sin\theta_p \\ Z_p = Z_r + \cos\theta_p \end{cases}$$

where the orthogonal coordinates of the origin of the spherical system are (X_r, Y_r, Z_r) .

3. A circle on the unit sphere of the spherical coordinate system has the following equation:

$$\cos\theta\cos\theta_0 + \cos(\phi - \phi_0)\sin\theta\sin\theta_0 = 1 - r^2/2.$$

where the spherical coordinates of the circle center are (ϕ_0, θ_0) and the radius of the circle is r .

4. The spherical equation of a circle which passes through three points (ϕ_1, θ_1) , (ϕ_2, θ_2) and (ϕ_3, θ_3) is

$$\begin{vmatrix} \cos\theta & \cos\phi\sin\theta & \sin\phi\sin\theta & 1 \\ \cos\theta_1 & \cos\phi_1\sin\theta_1 & \sin\phi_1\sin\theta_1 & 1 \\ \cos\theta_2 & \cos\phi_2\sin\theta_2 & \sin\phi_2\sin\theta_2 & 1 \\ \cos\theta_3 & \cos\phi_3\sin\theta_3 & \sin\phi_3\sin\theta_3 & 1 \end{vmatrix} = 0$$

5. If the spherical coordinates of two spatial points (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) are (ϕ_1, θ_1) and (ϕ_2, θ_2) respectively, then the straight line which passes through the two spatial points has the following spherical equation:

$$\begin{vmatrix} \cos\phi\sin\theta & \sin\phi\sin\theta & \cos\theta \\ \cos\phi_1\sin\theta_1 & \sin\phi_1\sin\theta_1 & \cos\theta_1 \\ \cos\phi_2\sin\theta_2 & \sin\phi_2\sin\theta_2 & \cos\theta_2 \end{vmatrix} = 0$$

6. The spherical coordinates of the mid-point of two points (ϕ_1, θ_1) and (ϕ_2, θ_2) are

$$\phi_{\text{mid}} = \arccos\left[\frac{(\cos\phi_1\sin\theta_1 - \cos\phi_2\sin\theta_2)}{\sqrt{\sin^2\theta_1 + \sin^2\theta_2 - 2\cos(\phi_1 - \phi_2)\sin\theta_1\sin\theta_2}}\right]$$

$$\theta_{\text{mid}} = \arccos\left[\frac{(\cos\theta_1 - \cos\theta_2)}{\sqrt{2 - 2[\cos\theta_1\cos\theta_2 + \cos(\phi_1 - \phi_2)\sin\theta_1\sin\theta_2]}}\right]$$

7. If the origin of a spherical coordinate system for the robot locates at (X_r, Y_r, Z_r) and a feature point A locates at (X_a, Y_a, Z_a) , the spherical coordinates (ϕ_a, θ_a) of the feature point A can be calculated by the following formula:

$$\phi_a = \arccos\left[\frac{(X_a - X_r)}{\sqrt{(X_a - X_r)^2 + (Y_a - Y_r)^2}}\right], (+0 \text{ or } +\pi)$$

$$\theta_a = \arccos[(Z_a - Z_r) / \sqrt{(X_a - X_r)^2 + (Y_a - Y_r)^2 + (Z_a - Z_r)^2}].$$

8. In the fixed orthogonal coordinate system, the equation of a plane which passes through the origin (X_r, Y_r, Z_r) of a spherical system and two points (ϕ_1, θ_1) and (ϕ_2, θ_2) which are related to the spherical coordinate system is

$$\begin{vmatrix} X & Y & Z & 1 \\ X_r & Y_r & Z_r & 1 \\ \cos\phi_1\sin\theta_1 & \sin\phi_1\sin\theta_1 & \cos\theta_1 & 0 \\ \cos\phi_2\sin\theta_2 & \sin\phi_2\sin\theta_2 & \cos\theta_2 & 0 \end{vmatrix} = 0$$

9. If the origins of two spherical coordinate systems for the robot are known as (X_{r1}, Y_{r1}, Z_{r1}) and (X_{r2}, Y_{r2}, Z_{r2}) and the corresponding spherical coordinates of a spatial feature point A are known as (ϕ_{a1}, θ_{a1}) and (ϕ_{a2}, θ_{a2}) respectively, the orthogonal coordinates of the feature point A can be calculated by the following formula:

$$\begin{cases} X_a = [(X_{r1}\text{tg}\phi_{a1} - X_{r2}\text{tg}\phi_{a2}) - (Y_{r1} - Y_{r2})] / (\text{tg}\phi_{a1} - \text{tg}\phi_{a2}), \\ Y_a = [(Y_{r1}\text{ctg}\phi_{a1} - Y_{r2}\text{ctg}\phi_{a2}) - (X_{r1} - X_{r2})] / (\text{ctg}\phi_{a1} - \text{ctg}\phi_{a2}), \\ Z_a = [(Z_{r1}\sin\phi_{a1}\text{tg}\theta_{a1} - Z_{r2}\sin\phi_{a2}\text{tg}\theta_{a2}) - (Y_{r1} - Y_{r2})] / (\sin\phi_{a1}\text{tg}\theta_{a1} - \sin\phi_{a2}\text{tg}\theta_{a2}). \end{cases}$$

10. The 3-D orthogonal coordinates of the origin of a spherical surface and the radius of the surface can be calculated with the information gathered from two views. The origin of the spherical surface is the intersection point of the following two straight lines:

$$X / (\cos\phi_o\sin\theta_o) = Y / (\sin\phi_o\sin\theta_o) = Z / \cos\theta_o$$

$$X / (\cos\phi_n\sin\theta_n) = Y / (\sin\phi_n\sin\theta_n) = Z / \cos\theta_n$$

where the (ϕ_o, θ_o) and (ϕ_n, θ_n) are the spherical coordinates of the centers of the two projections of the spherical surface in the old and new views. The radius of the spherical surface is

$$R = r_o D$$

where D is the distance between the old viewpoint and the origin of the spherical surface and r_o is the radius of the projection of the surface in the old view.

11. The axis of a cylindrical or a conical surface can be determined from two views. Assuming A and C are two 2s junctions and B and D are two corresponding 1aX (X= 1 or 2) junctions, then the axis is at the plane which passes through the viewpoint and the two midpoints of AB and CD. From two views, we can get two such planes that pass through the axis. Thus the axis lies on the intersection line of the two planes.

Appendix H

Rules for Solving Local Ambiguities

The rules for planning new viewpoints in 3-D to solve local ambiguities are set out below. These rules are derived from local geometrical properties, such as junction types and surface categories. [Note: Based on the Chakravarty [16] and Waltz [101] labeling schemes, a modified and extended labeling scheme shown in Appendix C has been used in our systems.]

1. If the type of a junction is 1w1, 1q, 2p1, 2p2, 2w2, 1v2 or 1v3, then the body containing the corresponding vertex touches the surface which is adjacent to the vertex.
2. If a portion of a surface wholly touches another surface, then this portion of the surface cannot be viewed, unless the robot removes the corresponding body away.
3. A 1a1 or 2s type junction indicates there is a round surface and thus the back part of the round surface is self-occluded. The new viewpoint should be located behind the corresponding body and in the plane formed by the limb and the old viewpoint.
4. A 1a2 type junction indicates there is a round surface, thus the back part of the round surface is self-occluded. The new viewpoint should be located (1) behind the corresponding body, (2) in the plane formed by the limb and the old viewpoint and (3) below the surface formed by the occluding line.
5. If two 2s junctions or two 1aX (X=1 or 2) junctions are line adjacent to each other, then the corresponding body has at least a part of a cylinder or cone, and the new viewpoint should be located behind the body.

6. A junction of the 11t1, 11t2, 11t3, 11t4, 21t1, 21t2, 21t3, 21t4, 21t5 or 21t6 type means there is an occlusion, and a new viewpoint should be placed along the shift of the t junction and far from the bar.
7. A 2w1 junction indicates that a surface bounded by the two arrow edges is self-occluded. The new viewpoint should be located in the other half space which is defined by the two arrow edges.
8. A 1v junction indicates that the two other surfaces which form the corresponding vertex are self-occluded. The new viewpoint should be located at the intersection of the two half spaces which are defined by the two edges of the junction and the old viewpoint respectively.
9. If an object or a part of an object which is currently interesting has edges cut by a viewport edge, then the new viewport should be extended in the direction that corresponds to the extension of the edge.

Appendix I

Knowledge for the CSG-EESI Conversion

To simulate the process by which humans perform the decomposition of bodies into their component parts, the following knowledge bases have been formed for the conversion system.

To limit the scope of this problem, some assumptions are made:

1. Bodies are formed by planar, circular cylindrical, circular conical and spherical faces. [Note: This assumption can be relaxed to include in the other types of faces when the expert system is expanded.]
2. No more than two quadric faces can intersect together in a body; this does not apply to planes. [Note: This assumption is due to the use of planes and extending concave quadric surfaces to separate subparts.]
3. A concave quadric face can not intersect with the other quadric faces. [Note: This assumption is not essential, it can be released when the expert system is expanded.]
4. Certain peculiar and pathological situations are not considered.

If a body does not meet these limitations, the conversion expert system will simply refuse to further decompose it.

The rules for the Controller are set out below. These rules are derived from the knowledge of geometry and are motivated by the need to find sensible methods to decompose solids into simple component parts.

1. If a body has at least one concave quadric face then:
 - a. if all of the faces connected to the concave quadric face are planes and all of them are convex-connected with the concave quadric face, then assign the decomposition task to the Micro_expert 1;
 - b. if all of the faces connected to the concave quadric face are planes and all of them are concave-connected with the concave quadric face, then assign the decomposition task to the Micro_expert 2;
 - c. if all of the faces connected to the concave face are planes and only some of them are convex-connected with the concave quadric face, then assign the decomposition task to the Micro_expert 3.
2. If a body has at least one convex quadric face then:
 - a. if all of the faces connected to the convex quadric face are planes and all of them are convex-connected with the convex quadric face, then assign the decomposition task to the Micro_expert 4;
 - b. if all of the faces connected to the convex quadric face are planes and all of them are concave-connected with the convex quadric face, then assign the decomposition task to the Micro_expert 5;
 - c. if all of the faces connected to the convex face are planes and only some of them are convex-connected with the convex quadric face, then assign the decomposition task to the Micro_expert 6;
 - d. If two convex quadric faces intersect with each other, then assign the decomposition task to the Micro-expert 7.
3. If a body consists of planes, then assign the decomposition task to the Micro_expert 8.

The rules for the Micro_experts are set out in turn:

Case 1 (Micro_expert 1) :

1. Find the intersection points between the concave quadric face and its convex-connected plane set C_a .
2. For each group of above edge-connected points, find 3 points such that a plane passing through them will leave the other points in the group on the same side of the plane as the quadric face.
3. Create a separating plane which is co-planar with the above plane, and find the connectivities for the separating plane.
4. Distinguish the following 2 sub-cases:
 - a. If a group of C_a planes corresponding to the current group of edge-connected points connects with a previous group, then the group of planes does not form a new subpart. Its corresponding separating plane will be added to the previously created subpart.
 - b. Otherwise, the group of planes with its connected faces, except the concave quadric face, and a separating plane create a new subpart of the body which forms a left-hand branch of the CSG tree.
5. After forming all left branches, the lowest right-hand branch will be found. It consists of the concave quadric surface, which changes its type to convex, and those separating planes.
6. Create a structural sub-tree corresponding to the above decomposition.
7. Further decompose the left branches.

Case 2 (Micro_expert 2) :

1. Find the intersection points between the concave quadric face and its concave-connected plane set C_b .
2. For each group of above edge-connected points, find 3 points such that a plane passing through them will leave the other points in the group on the same side of the plane as the quadric face.

3. Create a pair of separating planes which are co-planar with the above plane, and find the connectivities for the separating planes.
4. If a group of Cb planes corresponding to the current group of edge-connected points are not all totally in the concave quadric surface, then distinguish the following 2 sub-cases:
 - a. If the group of Cb planes connects with a previous group, then the group of planes does not form a new subpart. Its corresponding separating plane will be added to the previously created subpart.
 - b. Otherwise, the group of Cb planes with its connected faces, except the concave quadric face, and a separating plane create a new subpart of the body which forms a left branch of the CSG tree.
5. After forming all left branches, the lowest right-hand branch will be found. It consists of the other one of those separating plane pairs, and the connected plane set Cb and the concave quadric surface which changes its type to convex.
6. Create a structural sub-tree corresponding to the above decomposition.
7. Further decompose the left branches.

Case 3 (Micro_expert 3) :

1. Find out those planes that are totally or half in the concave quadric surface.
2. For those half-in planes, except the convex-connected planes, find the intersection points between them and the totally-in planes.
3. In the intersection point group, find 3 points such that a plane passing through them will leave the other points in the group on the same side of the plane as the quadric face.
4. Create a separating plane which is co-planar with the above plane, and find the connectivities for the separating plane.

5. The quadric surface, which now changes its type to convex, with the totally-in and half-in planes forms a set of subparts, each of which consists of connected planes.
6. $\text{DIFFERENCE}\{ [\text{original surface set}], \text{UNION}\{[\text{totally-in planes}], [\text{the quadric surface}]\}$ form the left subpart called *Part_1*.
7. The part of CSG tree has the following like structure: $\text{DIFFERENCE}(\text{Part}_1, \text{DIFFERENCE}(\text{the quadric surface, the set of subparts in step 5}))$.
8. Further decompose the left branch.
9. Further decompose the set of subparts in step 7.

Case 4 (Micro_expert 4) :

1. If all convex-connected planes are out of the convex quadric surface and the other faces do not connect with the quadric face, then:
 - a. find the intersection points between the convex quadric face and its convex-connected plane set C_a ;
 - b. for each group of above edge-connected points, find 3 points such that a plane passing through them will leave the other points in the group on the same side of the plane as the quadric face;
 - c. create a pair of separating planes which are co-planar with the above plane, and find the connectivities for the separating planes;
 - d. the group of planes with its connected faces, except the convex quadric face, and a separating plane create a new subpart of the body which forms a left branch of the CSG tree;
 - e. the plane group and the other separating plane with the remaining faces form the right branch;
 - f. further decompose the two branches.

2. If all C_a planes are in the quadric surface and the other faces are planes and in the quadric surface too, then if the total number of planes of the part is less than 7 or if there at most exist one concave edge, then they all form an EESI model; otherwise, separate the part as follows:

- a. Find the intersection points between the convex quadric face and its convex-connected plane set C_a .
- b. For each group of above edge-connected points, find 3 points such that a plane passing through them will leave the other points in the group on the same side of the plane as the quadric face.
- c. Create a pair of separating planes which are co-planar with the above plane, and find the connectivities for the separating planes.
- d. The group of planes with its connected faces, except the concave quadric face, and a separating plane create a new subpart of the body which forms a left branch of the CSG tree.
- e. The plane group and the other separating plane with the remaining faces form the right branch.
- f. Further decompose the two branches.

3. If there are some C_a planes which cross-intersect the convex quadric surface, then each of these connecting planes with the convex quadric face belongs to the right branch and those connected faces which connect with the plane form left branches. Separating planes which are co-planar with the plane will be created for each left branch. Further decompose those branches.

Case 5 (Micro_expert 5) :

1. Find the intersection points between the convex quadric face and its concave-connected plane set C_b .
2. If there are some planes which cross-intersect the convex quadric surface, then each of these connecting planes with their connected

faces, except the convex quadric face, form a left branch and the remaining faces form the right branch. Separating planes which are co-planar with the connecting planes will be created for the right branch. Further decompose these two branches.

3. Otherwise, find those connected plane groups from C_b . For each plane group, create separating planes which are co-planar with these planes and have reverse normal directions. The plane group with their connected faces, except the quadric face, form the left branches, and the separating planes with the remaining faces form the right branch. Further decompose these two branches.

Case 6 (Micro_expert 6) :

1. Find the intersection points between the convex quadric face and its convex & concave connected plane set C_a .
2. For each group of above edge-connected points, find 3 points such that a plane passing through them will leave the other points in the group on the same side of the plane as the quadric face.
3. Distinguish the following 3 sub-cases:
 - a. If the current group only consists of convex-connected planes, then transfer control to Case 4.
 - b. If the current group only consists of concave-connected planes, then transfer control to Case 5.
 - c. Otherwise, find those intersection points P_a in the group which are the boundary points of the concave-connected planes. From these points P_a , find 3 points (if no more than 2 P_a points are in the group, then select the other one from the group) which form a separating plane such that the inner-product between its normal and the normal of the plane formed in step 2 should be positive, and it leave the other points of P_a on the normal positive side of the separating plane.

Create another separating plane which is co-planar with the above

one and has the reverse normal. Find the connectivities for the two separating planes.

4. Corresponding to each group of edge-connected points in step 2, create a left branch which consists of a separating plane, the group of convex-connected planes and their connected faces except the quadric face. After forming all left branches, the lowest right-hand branch will be found. It consists of the other separating planes, the connected plane sets Cab and the different set between the original face set of the body and the face sets of the left branches.
5. Form a part of structural tree corresponding to this decomposition.
6. Further decompose each left branch.

Case 7 (Micro_expert):

Here we only consider the following sub-cases:

1. A spherical face connects with other spherical face. In this case, the first spherical face with its connected faces, except the second spherical face, form the left part, the remaining faces form the right part. Then further decompose the two parts.
2. A spherical face connects with a cylindrical face. In this case, if the axis of the cylindrical surface passes through the center of the spherical surface, then it is necessary to create a separating plane which is perpendicular to the axis and its distance to the center of the spherical surface can be determined.

The spherical face with its connected faces, except the cylindrical face, form the left part. The cylindrical face with its connected faces, except the spherical face, and the separating plane form the right part.

Further decompose the two parts.

3. A spherical face connects with a conical face. If the axis of the conical surface passes through the center of the spherical surface, this case is like the previous case, except the distance from the separating plane to the center of the spherical surface has a different value.

4. A cylindrical face connects with another cylindrical face.
 - a. If their axes are parallel, then create a pair of separating planes at their intersection. Each cylindrical face with its corresponding separating surface and its connected faces, except the another cylindrical face, form a part of the body. Then further decompose the two parts.
 - b. If their radii are same and their axes intersect with each other, then create a pair of separating planes at their intersection. Each cylindrical face with its corresponding separating surface and its connected faces, except the other cylindrical face, form a part of the body. Then further decompose the two parts.

Case 8 (Micro_expert 8) :

1. If the total number of planes in the body is less than 8, then it is a simple body and it corresponds to an EESI model.
2. If a plane F has at least two separated concave-connected plane groups, then do the following:
 - a. If a plane group consists of at least two planes, create a separating plane, which is co-planar with the plane F. Decompose the body into parts according to the following method:
 - i. If the plane group is the subset of a previously created subpart, then it is only necessary to add the corresponding separating plane to the subpart.
 - ii. Otherwise, create a new subpart for the plane group. The separating plane will belong to the subpart. If a plane belongs to the group, then test the plane.
 1. If the plane is totally on one side of the plane F, then:
 - a. if it is on the normal positive side of plane F, the plane totally belongs to the subpart and its

connected planes will be added to the plane group for further test:

- b. if it is on the negative side, then the plane belongs to the main-part which includes plane F, and it will be deleted from the group.
 2. Otherwise, the plane will be deleted from the group: it belongs to both the subpart and the main-part.
- b. If a plane group only consists of one plane R and, to R, F with other planes forms a concave-connected plane group, then create a separating plane, which is co-planar with the plane R. Decompose the body into parts according to the following method:
 - i. If plane R belongs to a previously created subpart, then it is only necessary to add the corresponding separating plane to the subpart.
 - ii. Otherwise, create a new subpart for the plane group which consists of plane R. First add the connected planes of R, except F, into the plane group. If a plane belongs to the group, then test the plane.
 1. If the plane is totally on one side of the plane R, then:
 - a. if it is on the normal negative side of plane R, the plane totally belongs to the subpart and its connected planes will be added to the plane group for further test;
 - b. if it is on the positive side, then the plane belongs to the main-part which includes plane F and the separating plane, and it will be deleted from the group.
 2. Otherwise, the plane will be deleted from the group: it belongs to both the subpart and the main-part.
- c. If a plane group consists only of one plane R and, for R, F also

itself forms a concave-connected plane group, then create a pair of separating planes which are co-planar with the bisector of planes F and R. Decompose the body into parts according to the following method:

- i. If plane R belongs to a previously created subpart, then it is only necessary to add a corresponding separating plane to the subpart and add the other separating plane to the main-part.
- ii. Otherwise, create a new subpart for the plane group which consists of plane R. The two separating planes will belong to the subpart and the main-part respectively. Add the connected planes of R, except F, into the plane group. If a plane belongs to the group, then test the plane.

1. If the plane is totally on one side of both plane F and the separating plane which with F belongs to the main-part, then:

- a. if it on the normal positive side of them, the plane totally belongs to the subpart and its connected planes will be added to the plane group for further test;
- b. if it on the negative side, then the plane belongs to the main-part which includes plane F and the separating plane, and it will be deleted from the group.

2. Otherwise, the plane will be deleted from the group; it belongs to both the subpart and the main-part.

The subpart forms a left branch of the structural tree, the main-part forms the right branch.

Further decompose the left branches.

Further decompose the right branch.

3. If a plane F only has one concave-connected plane group and some planes in the group are convex-connected with each other, then create a separating plane which is co-planar with F, and decompose the body into parts according to the following method:

Create a new subpart for the plane group. The separating plane will belong to the subpart. If a plane belongs to the group, then test the plane.

- a. If the plane is totally on one side of the plane F, then:
 - i. if it is on the normal positive side of plane F, the plane totally belongs to the subpart and its connected planes will be added to the plane group for further test;
 - ii. if it is on the negative side, then the plane belongs to the main-part which includes plane F, and it will be deleted from the group.
- b. Otherwise, the plane will be deleted from the group; it belongs to both the subpart and the main-part.

The subpart forms a left branch of the structural tree, the main-part forms the right branch.

Further decompose the left branches.

Further decompose the right branch.

4. Otherwise, the planar body forms an EESI model.

Appendix J

The Input and Output Forms of a Test Body for the CSG-EESI Conversion System

```

/*      The input form of body1 is as follows:      */
body(body1, [f1, f2, f3, f4, f5, f6, f7, f9, f10, f11, f12,
            f13], [], [f8]).

face(f1, p, [0, 0, 1], 10, [f3, f4, f5, f6, f7, f8, f9, f10,
                            f11, f12, f13], [], [0, 0, 0], [150, 50, 10]).

face(f2, p, [0, 0, -1], 0, [f3, f4, f5, f6, f7, f8, f9, f10,
                            f11, f12, f13], [], [0, 0, 0], [150, 50, 0]).

face(f3, p, [0, 1, 0], 50, [f1, f2, f4, f6], [], [0, 0, 0],
    [150, 50, 0]).

face(f4, p, [1, 0, 0], 150, [f1, f2, f3, f5], [], [0, 0, 0],
    [150, 50, 0]).

face(f5, p, [0, -1, 0], 50, [f1, f2, f4, f7], [], [0, 0, 0],
    [150, -50, 0]).

face(f6, p, [-0.71, -0.71, 0], 0, [f1, f2, f3, f12], [], [0,
    0, 0], [0, 0, 0]).

face(f7, p, [-0.71, 0.71, 0], 0, [f1, f2, f5, f13], [], [0,
    0, 0], [-50, -50, 0]).

face(f8, bcyl18, [0, 0, 1], 40, [f1, f2], [f9, f13], [0, 0,
    0], [20, 0, 1]).

face(f9, p, [1, 0, 0], 0, [f1, f2], [f8, f10], [0, 0, 0], [0,

```

¹⁸This indicates a concave cylindrical surface.

20, 0]).

face(f10, p, [0, 1, 0], 8, [f1, f2, f11], [f9], [0, 0, 0],
[8, 8, 0]).

face(f11, p, [1, 0, 0], 8, [f1, f2, f10, f12], [], [0, 0, 0],
[8, 8, 0]).

face(f12, p, [0, -1, 0], 0, [f1, f2, f11, f6], [], [0, 0, 0],
[0, 0, 0]).

face(f13, p, [1, 0, 0], 5, [f1, f2, f7], [f8], [0, 0, 0],
[-5, -5, 0]).

point(p1, [f1, f3, f6], [-50, 50, 10]).

point(p2, [f1, f3, f4], [150, 50, 10]).

point(p3, [f1, f4, f5], [150, -50, 10]).

point(p4, [f1, f5, f7], [-50, -50, 10]).

point(p5, [f1, f6, f12], [0, 0, 10]).

point(p6, [f1, f11, f12], [8, 0, 10]).

point(p7, [f1, f11, f10], [8, 8, 10]).

point(p8, [f1, f10, f9], [0, 8, 10]).

point(p9, [f1, f8, f9], [0, 20, 10]).

point(p10, [f1, f8, f13], [-5, -19.36, 10]).

point(p11, [f2, f3, f6], [-50, 50, 0]).

point(p12, [f2, f3, f4], [150, 50, 0]).

point(p13, [f2, f4, f5], [150, -50, 0]).

point(p14, [f2, f5, f7], [-50, -50, 0]).

point(p15, [f2, f6, f12], [0, 0, 0]).

point(p16, [f2, f11, f12], [8, 0, 0]).

```

point(p17, [f2, f11, f10], [8, 8, 0]).
point(p18, [f2, f10, f9], [0, 8, 0]).
point(p19, [f2, f8, f9], [0, 20, 0]).
point(p20, [f2, f8, f13], [-5, -19.36, 0]).
point(p21, [f1, f7, f13], [-5, -5, 10]).
point(p22, [f2, f7, f13], [-5, -5, 0]).

/* The printout of the system in analyzing body1 is as follows: */

```

"Now the system analyses the 'body1'.

CPU time since C-PROLOG was started is '47.6166' seconds."

" A new separating plane is created as:

```

'vector(sep1, sep, [-0.707107, 0.707107, 0], 0, [f7.f1.f2], [f6], _7275 , _7276,
[0.0.0], [0.0.10])' "

```

'subpart1'¹⁹ has been decomposed as the following structure:

```

'[[sep1.f1.f2.f3.f4.f5.f6.f7], [uni.void.void]]' "

```

'body1' has been decomposed as the following structure:

```

'[[f8, f1, f2, f3, f4, f5, f6, f7, f9, f10, f11, f12, f13], [dif, [[sep1, f1, f2, f3, f4,
f5, f6, f7], [uni, void, void]], [[rf8,20 f13, f12, f11, f10, f9, f7, f6, f2, f1], [dif,
[[rf8], [uni, void, void]], [[f7, f13, rf8, f6, f2, f1, f12, f11, f10, f9], [uni, [[rf8, f6,

```

¹⁹subpart1 is the leftmost leaf of the decomposed CSG tree.

²⁰Here rf8 is a convex quadric surface corresponding to f8.

f2, f1, f12, f11, f10, f9], [uni, void, void]], [[rf8, f7, f2, f1, f13], [uni, void, void]]]]]]]]]]]

"CPU time since C-PROLOG was started is '106.433' seconds. "

References

1. Aggarwal, A. *The Art Gallery Theorem: Its Variations, Applications and Algorithmic Aspects*. Ph.D. Th., Dept. of Electrical Engineering and Computer Science, The John Hopkins University, 1984.
2. Agin, G.J. Hierarchical representation of three-dimensional objects using verbal models. Note 182, Artificial Intelligence Center, SRI International, March, 1979.
3. Ahuja, N., R.T. Chien, R. Yen, and N. Bridwell. Interference detection and collision avoidance among three dimensional objects. Proc. of the First Annual National Conference on A.I., Aug., 1980, pp. 44-48.
4. Asada, M. *Understanding of three-dimensional motions in trihedral world*. Ph.D. Th., Dept. of Control Engineering, Osaka University, Japan, Jan. 1982.
5. Asano, T. Efficient algorithms for finding the visibility polygon for a polygonal region with holes. personal communication, 1984.
6. Avis, D. and G.T. Toussaint. "An efficient algorithm for decomposing a polygon into star-shaped polygons". *Pattern Recognition* 13 (1981), 395-398.
7. Badler, N. Three-dimensional motion from two-dimensional picture sequences. Proc. of 2nd International Joint Conference on Pattern Recognition, Copenhagen, Denmark, Aug., 1974.
8. Ballard, D.H. and C.M. Brown. *Computer Vision*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.
9. Barrow, H.G. and J.M. Tenenbaum. "Interpreting line drawing as three-dimensional surfaces". *Artificial Intelligence* 17 (1981), 75-116.
10. Bhanu, B. "Representation and shape matching of 3-D objects". *IEEE Trans. on PAMI* PAMI-6(3) (May 1984), 340-351.
11. Bhattacharya, B.K. and H.A. ElGindy. An efficient algorithm for an intersection problem with an Application. TR 85-4, Simon Fraser University, May, 1985.

12. Binford, T.O. Visual perception by computer. Proc. IEEE Conf. System and Control, Miami, Fla., 1971.
13. Binford, T.O. "Inferring surfaces from images". *Artificial Intelligence* 17 (1981), 205-244.
14. Brooks, R.A. "Symbolic reasoning among 3-D models and 2-D images". *Artificial Intelligence* 17 (1981), 285-348.
15. Brou, P. "Using the Gaussian image to find the orientation of objects". *The International Journal of Robotics Research* 3 (1984), 89-125.
16. Chakravarty, I. "A generalized line and junction labeling scheme with applications to scene analysis". *IEEE Trans. on PAMI PAMI-1* (1979), 202-205.
17. Chandransekaran, B., et al. An approach to medical diagnosis based on conceptual structures. Proc. of the IJCAI, Tokyo, Japan, 1979, pp. 134-142.
18. Charniak. *Introduction to Artificial Intelligence*. Addison-Wesley Publishing Company, 1985.
19. Chazelle, B. and D. Dobkin. Decomposing a polygon into its convex parts. Proc. of the 11th Annual ACM Symposium on Theory of Computing, 1979, pp. 38-48.
20. Christofides, N. The travelling salesman problem - a survey. In R. J. Wilson, Ed., *Applications of Combinatorics*, Shiva Publishing Limited, 1982, pp. 29-49.
21. Chvatal, V. "A combinatorial theorem in plane geometry". *J. Combinatorial Theory* 18 (1975), 39-41.
22. Clowes, M.B. "On seeing things". *Artificial Intelligence* 2 (1971), 79-112.
23. Connolly, C.I. The determination of next best views. Proc. IEEE Inter. Conf. Robotics and Automation, St. Louis, Miss., 1985, pp. 432-435.
24. Crowley, J.L. Dynamic world modeling for an intelligent mobile robot using a rotating ultra-sonic ranging device. Proc. IEEE Inter. Conf. Robotics and Automation, St. Louis, Miss., 1985, pp. 128-137.
25. Edelsbrunner, H., L.J. Guibas and J. Stolfi. Optimal point location in a monotone subdivision. 2. Systems Research Center, Digital Equipment Corporation, Palo Alto, CA., 1984.

26. ElGindy, H. and D. Avis. "A linear algorithm for computing the visibility polygon from a point". *J. Algorithms* 2 (1981), 186-197.
27. Floyd, R.W. "Algorithm 97: Shortest Path". *Comm. ACM* 5 (1962), 345.
28. Freeman, H. and P.P. Lourel. "An algorithm for the solution of the two-dimensional 'hidden-line' problem". *IEEE Trans. on Electronic Computer EC-16* (1967), 784-790.
29. Ganapathy, S. *Reconstruction of scenes containing polyhedra from stereo pair of views*. Ph.D. Th., Stanford Artificial Intelligence Laboratory, Memo AIM-272, December 1975.
30. Georgeff, M.P. "Procedural control in production systems". *Artificial Intelligence* 18 (1982), 175-201.
31. Gilmore, J.F. and K. Pulaski. A survey of expert system tools. Proc. of the Second Conference on AI Application, Miami Beach, Florida, 1985, pp. 498-502.
32. Giralt, G., R. Sobek, and R. Chatila. A multi-level planning and navigation system for a mobile robot; a first approach to HILARE. Proc. of the 6th IJCAI, Tokyo, Japan, Aug., 1979, pp. 335-338.
33. Guzman, A. Decomposition of a visual scene into three-dimensional bodies. Proc. AFIPS Fall Joint Compt. Conf., Vol. 33, 1968, pp. 291-304.
34. Hanson, A. and E. Riseman. VISIONS, A computer system for interpreting scenes. In A. Hanson and E. Riseman, Ed., *Computer Vision Systems*, Academic Press, 1978, pp. 303-333.
35. Harmon, P. and D. King. *Expert Systems*. John Wiley & Sons, Inc., New York, 1985.
36. Henderson, T.C. "Efficient 3-D object representations for industrial vision systems". *IEEE Trans. on PAMI PAMI-5(6)* (Nov. 1983), 609-618.
37. Herman, M., T. Kanade, and S. Kuroe. "Incremental Acquisition of a Three-Dimensional scene model from images". *IEEE Trans. on PAMI PAMI-6(3)* (May 1984), 331-340.
38. Herman, M. Matching three-dimensional symbolic description obtained from multiple views of a scene. Proc. IEEE Conf. Computer Vision and Pattern Recognition, San Francisco, Ca., 1985, pp. 585-590.

39. Horn, B.K.P. "Extended Gaussian Images". *Proceedings of the IEEE* 72 (Dec. 1984), 1671-1686.
40. Huffman, D.A. Impossible objects as nonsense sentence. In B. Meltzer and D. Michie, Ed., *Machine Intelligence, Vol. 6*, Edinburgh Univ. Press, Edinburgh, U.K., 1971.
41. Ikeuchi, K. Recognition of objects using the extended Gaussian image. Proc. of IJCAI-81, Vancouver, B.C., Canada, Aug., 1981, pp. 595-600.
42. Iyengar, S.S., C.C. Jorgensen, S.V.N. Rao, and C.R. Weisbin. Learned navigation paths for a robot in unexplored terrain. Proc. of the Second Conference on A.I. Applications, Miami Beach, Florida, Dec., 1985, pp. 148-155.
43. Kahn, J., M. Klawe, and D. Kleitman. "Traditional Galleries Require Fewer Watchmen". *SIAM J. of Algebraic and Discrete Methods* 4 (1983), 194-206.
44. Keil, J.M. *Decomposing Polygons into Simpler Components*. Ph.D. Th., University of Toronto, April 1983.
45. Kim, H.-S. R.C. Jain, and R.A. Volz. Object recognition using multiple views. Proc. IEEE Inter. Conf. Robotics and Automation, St. Louis, Miss., 1985, pp. 28-33.
46. Koutsou, A. A survey of model-based robot programming languages. DAI Working Paper No. 108, Dept. of A.I., Univ. of Edinburgh, Dec., 1981.
47. Lavin, M.A. *Analysis of scenes from a moving viewpoint*. Ph.D. Th., MIT AI Laboratory, 1977.
48. Lee, D.T. and F.P. Preparata. "An optimal algorithm for finding the kernel of a polygon". *JACM* 26 (1979), 415-421.
49. Lee, D.T. and C.C. Yang. "Location of Multiple points in a planar subdivision". *Information Processing Letters* 9 (1979), 190-193.
50. Lee, H.C. and K.S. Fu. A computer vision system for generating object description. Proc. of the IEEE Conference on Pattern Recognition and Image Processing, Las Vegas, Nevada, June, 1982, pp. 466-472.
51. Lee, D.T. "Visibility of a simple polygon". *Computer Vision, Graphics and Image Processing* 22 (1983), 207-221.

52. Levine, M.D. and S.I. Shaheen. "A modular computer vision system for picture segmentation and interpolation". *IEEE Trans. PAMI PAMI-3* (Sept. 1981), 540-556.
53. Little, J.J. An iterative method for reconstructing convex polyhedra from extended Gaussian images. Proc. Nat. Conf. on Artificial Intelligence, Washington, D C, Aug., 1983, pp. 247-254.
54. Lowe, D.G. and T.O. Binford. Interpreting of geometric structure from image boundaries. SPIE 281 Techniques and Applications of Image Understanding, 1981, pp. 224-231.
55. Lozano-Perez, T. and M.A. Wesley. "An Algorithm for planning collision-free paths among polyhedral obstacles". *Communications of the ACM 22* (1979), 560-570.
56. Lozano-Perez, T. "Automatic planning of manipulator-transfer movements". *IEEE Trans. on Systems, Man and Cybernetics SMC-11* (Oct. 1981), 681-698.
57. Lozano-Perez, T. "Spatial planning: a configuration space approach". *IEEE Trans. on Computers C-32* (Feb. 1983), 108-120.
58. Mackworth, A.K. "Interpreting pictures of polyhedral scenes". *Artificial Intelligence 4* (1973), 121-137.
59. Marr, D. and H.K. Nishihara. Representation and recognition of the spatial organization of three dimensional structure. Proc. R. Soc. London, 1978, pp. 269-294.
60. McCain, H.G. A hierarchically controlled, sensory interactive robot in the automated manufacturing research facility. Proc. of the IEEE International Conference on Robotics and Automation, St. Louis, Missouri, March, 1985, pp. 931-941.
61. McPherson, C.A., J.B.K. Tio, F.A. Sadjachi and E.L. Hall. Curved surface representation for image recognition. Proc. of the IEEE Conference on Pattern Recognition and Image Processing, Las Vegas, Nevada, June, 1982, pp. 363-369.
62. Nau, D.S. Expert computer systems: A tutorial. TR-1201, University of Maryland, Aug., 1982.
63. Newell, A. and H.A. Simon. *Human Problem Solving*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1972.

64. Nilsson, N.J. A mobile automaton: an application of artificial intelligence techniques. *Proc. IJCAI*, 1969, pp. 509-520.
65. Nitzan, D. "Development of intelligent robots: Achievements and Issues". *IEEE Journal of Robotics and Automation RA-1* (March 1985), 3-13.
66. O'Rourke, J. and K.J. Supowit. "Some NP-Hard Polygon Decomposition Problems". *IEEE Trans. on Information Theory IT-29* (March 1983), 181-190.
67. Perkins, W.A. Model-based vision system for scenes containing multiple parts. *Proc. of the 5th IJCAI, M.I.T.*, 1977, pp. 678-684.
68. Pogorelov, A.V. "Extrinsic geometry of convex surfaces". *Translations of Mathematical Monographs 35* (1973), 133-139.
69. Pogorelov, A.V.. *The Minkowsky multidimensional problem*. New York: Winston, 1978.
70. Preiss, K. "Algorithms for automatic conversion of a 3-view drawing of a plane-faced part to the 3-D representation". *Computers in Industry 2* (1981), 133-139.
71. Requicha, A.A.G. Representations of rigid solid objects. In Encarnacao, J., Ed., *Lecture Notes in Computer Science 89 : Computer Aided Design Modelling, Systems Engineering, CAD-Systems*, Springer-Verlag Berlin Heidelberg New York, 1980, pp. 2-78.
72. Requicha, A.A.G. "Representations for rigid solids : theory, methods, and systems". *ACM Computing Surveys 12* (Dec. 1980), 437-464.
73. Requicha, A.A.G. and H.B. Voelcker. "Solid modeling: current status and research directions". *IEEE Computer Graphics & Applications 3, 7* (Oct. 1983), 25-37.
74. Requicha, A.A.G. "Toward a theory of geometric tolerancing". *The Int. J. Robotics Res. 2, 4* (1983), 45-60.
75. Requicha, A.A.G. and S.C. Chan. "Representation of geometric features, tolerances, and attributes in solid modelers based on constructive geometry". *IEEE Journal of Robotics and Automation RA-2, 3* (Sept. 1986), 156-166.
76. Roac, J.W. and J.K. Aggarwal. "Computer tracking of objects moving in space". *IEEE Trans. PAMI PAMI-1* (1979), 127-134.

77. Roberts, L.G. Machine Perception of three dimensional solids. In J.T. Tippett, et al., Ed., *Optical and Electrooptical Information Processing*, MIT Press, 1965.
78. Rueb, K.D. and A.K.C. Wong. A hypergraph representation of free space for path planning. Proc. of the IEEE 1985 Conference on Computer vision and Pattern Recognition, San Francisco, California, 1985, pp. 184-188.
79. Sacerdoti, E.D. Plan generation and execution for robotics. Technique Note 290, SRI international, April, 1980.
80. Sakurai, H. and D.C. Gossard. "Solid model input through orthographic views". *ACM Computer Graphics* 17, 3 (1983), 243-249.
81. Schachter, B. "Decomposition of polygon into convex sets". *IEEE Trans. on Computers* C-27, 11 (Nov. 1978), 1078-1082.
82. Shapira, R. Reconstruction of curved-surface bodies from a set of imperfect projections. Proc. of the 5th IJCAI, 1977, pp. 628-634.
83. Shapira, R. and H. Freeman. "A cyclic-order property of bodies with three-face vertices". *IEEE Trans. on Computers* C-26, 10 (Oct. 1977), 1035-1039.
84. Shapira, R. and H. Freeman. "Computer description of bodies bounded by quadric surfaces from a set of imperfect projections". *IEEE Trans. on Computers* C-27, 9 (Sept. 1978), 841-854.
85. Shapiro, L.G. and J.D. Moriarty. Sticks, plates, and blobs : a three-dimensional object representation for scene analysis. Proc. of the First Annual National Conf. on A.I., 1980, pp. 28-30.
86. Shapiro, L.G. and R. M. Haralick. Matching three-dimensional models. Proc. of IEEE 1981 Pattern Recognition and Image Processing Conf., Dallas, Texas, 1981, pp. 534-541.
87. Shapiro, L.G. The role of AI in computer vision. Proc. of the Second Conference on A.I. Applications, Miami Beach, Florida, Dec., 1985, pp. 76-81.
88. Shirai, Y. "Robot Vision". *Future Generations Computer Systems* 1 (Sept. 1985), 325-352.
89. Shirai, Y. Key Issues of Robotics Research. In H. Hanafusa and H. Inoue, Ed., *Robotics Research, The Second Int. Symposium*, MIT Press, 1985, pp. 505-510.

90. Smith, D.A. Using enhanced spherical images for object representation. A. I. Memo 530, M. I. T., May, 1979.
91. Suri, S. and J. O'Rourke. Worst-case optimal algorithms for constructing visibility polygons with holes. Proc. of the Second Annual ACM Symposium on Computational Geometry, New York, June, 1986, pp. 14-23.
92. Tenenbaum, J.M. and H.G. Barrow. "Experiments in interpretation - guided segmentation". *Artificial Intelligence* 8, 3 (1977), 241-274.
93. Thompson, A.M. The navigation system of the JPL robot. Proc. of the 5th IJCAI, M.I.T., 1977, pp. 749-757.
94. Tsugawa, S., et al. An automobile with artificial intelligence. Proc. of the 6th IJCAI, Tokyo, 1979, pp. 893-895.
95. Tsuji, S. Monitoring of a building environment by a mobile robot. In H. Hanafusa and H. Inoue, Ed., *Robotics Research, The Second Int. Symposium*, MIT Press, 1985, pp. 349-356.
96. Turchan, M.P. and A.K.C. Wong. Low-level learning for a mobile robot: Environment model acquisition. Proc. of the Second Conference on A.I. Applications, Miami Beach, Florida, Dec., 1985, pp. 156-161.
97. Turner, K.J. Computer perception of curved objects. presented at the AISB Summer Conf., Univ. Sussex, Brighton, England, 1975.
98. Udupa, S.M. Collision detection and avoidance in computer controlled manipulators. Proc. of the 5th IJCAI, M. I. T., 1977, pp. 737-748.
99. Underwood, S.A. and C.L. Coates. "Visual learning from multiple views". *IEEE Trans. on Computers C-24*, 6 (June 1975), 651-661.
100. Vossler, D.L. "Sweep-to-CSG conversion using pattern recognition techniques". *IEEE CG & A* 5 (Aug. 1985), 61-68.
101. Waltz, D.L. Understanding line drawings of scenes with shadows. In P. Winston, Ed., *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975, pp. 19-91.
102. Weiss, L.E., A.C. Sanderson, and C.P. Neuman. Dynamic visual servo control of robots: An adaptive image-based approach. Proc. of IEEE Int. Conf. on Robotics and Automation, St. Louis, Missouri, March, 1985, pp. 662-669.

103. Wesley, M.A. and G. Markowsky. Generation of solid models from two-dimensional and three-dimensional data. In M. S. Pickett and J. W. Boyse, Ed., *Solid Modeling by Computers: From Theory to Applications*, Plenum Press, 1984, pp. 23-50.