1-1-2003

Three dimensional pattern recognition using feature-based indexing and rule-based search

Jae-Kyu Lee
University of Nevada, Las Vegas

THREE DIMENSIONAL PATTERN RECOGNITION USING FEATURE-BASED INDEXING AND RULE-BASED SEARCH

by

Jae-Kyu Lee

M. Sc., The University of Florida, Gainesville, Florida, U. S. A, 1998

Researcher, Agency for Defense Development, South Korea, 1995

M. Sc., Kon-Kuk University, South Korea, 1990

B. Sc., Kon-Kuk University, South Korea, 1988

A thesis submitted in requirements
for the partial fulfillment of the

Doctor of Philosophy Degree in Mechanical Engineering
Department of Mechanical Engineering
College of Engineering

Graduate College
University of Nevada Las Vegas
December 2003

UNLV

Dissertation Approval
The Graduate College
University of Nevada, Las Vegas

Dec. 17th , 20 03

The Dissertation prepared by

Jae-Kyu Lee

Entitled

Three Dimensional Pattern Recognition using Feature-Based Indexing and Rule-Based Search

is approved in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Examination Committee Chair

Examination Committee Member

Examination Committee Member
~~Dean of the Graduate College~~

Examination Committee Member

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

Dean of the Graduate College

ABSTRACT

Three Dimensional Pattern Recognition using Feature-Based Indexing and Rule-Based Search

by

Jae-Kyu Lee

Dr. Georg Mauer, Examination Committee Chair
Professor of Mechanical Engineering
University of Nevada, Las Vegas

In flexible automated manufacturing, robots can perform routine operations as well as recover from atypical events, provided that process-relevant information is available to the robot controller. Real time vision is among the most versatile sensing tools, yet the reliability of machine-based scene interpretation can be questionable. The effort described here is focused on the development of machine-based vision methods to support autonomous nuclear fuel manufacturing operations in hot cells.

This thesis presents a method to efficiently recognize 3D objects from 2D images based on feature-based indexing. Object recognition is the identification of correspondences between parts of a current scene and stored views of known objects, using chains of segments or *indexing vectors*. To create indexed object models, characteristic model image features are extracted during preprocessing. Feature vectors representing model object contours are acquired from several points of view around each object and stored. Recognition is the process of matching stored views with features or patterns detected in a test scene.

Two sets of algorithms were developed, one for preprocessing and indexed database creation, and one for pattern searching and matching during recognition. At recognition time, those indexing vectors with the highest match probability are retrieved from the model image database, using a nearest neighbor search algorithm. The nearest neighbor search predicts the best possible match candidates. Extended searches are guided by a search strategy that employs knowledge-base (KB) selection criteria. The knowledge-based system simplifies the recognition process and minimizes the number of iterations and memory usage.

Novel contributions include the use of a feature-based indexing data structure together with a knowledge base. Both components improve the efficiency of the recognition process by improved structuring of the database of object features and reducing data base size. This data base organization according to object features facilitates machine learning in the context of a knowledge-base driven recognition algorithm. Lastly, feature-based indexing permits the recognition of 3D objects based on a comparatively small number of stored views, further limiting the size of the feature database.

Experiments with real images as well as synthetic images including occluded (partially visible) objects are presented. The experiments show almost perfect recognition with feature-based indexing, if the detected features in the test scene are viewed from the same angle as the view on which the model is based. The experiments also show that the knowledge base is a highly effective and efficient search tool recognition performance is improved without increasing the database size requirements. The experimental results indicate that feature-based indexing in combination with a

knowledge-based system will be a useful methodology for automatic target recognition (ATR).

TABLE OF CONTENTS

ABSTRACT iii

LIST OF FEAGURES viii

GLOSSARY x

ACKNOWLEDGEMENTS xi

CHAPTER1 INTRODUCTION 1
Transmutation Fuel Process 1
Introducing Indexing Method 6
Problems of Indexing 9
Feature-Based Indexing and Knowledge-Based System 11
Thesis Organization 13

CHAPTER2 INVARIANT INDEXING METHOD AND PREVIOUS WORK 14
The Invariant Indexing Method 14
Building a Hash Table and Complexity Analysis 20
Related Work 23
Limitations of Indexing Method 25

CHAPTER3 PREPROCESSING BY FEATURE-BASED CONCEPT 28
Previous Work on Feature-Based Indexing 28
Definitions of Attribute in Feature-Based Modeling 31
Probabilistic Similarity Analysis 39
Summary of Feature-Based Geometry Modeling Concept 47

CHAPTET4 A KNOWLEDGE BASE FOR OBJECT RECOGNITION 49
Introducing the Knowledge-Based System 46
Related Work 51
Expert System Implementation: CLIPS 53
Rules and Decision Functions for Probabilistic Reasoning 54
Occlusion 60
Assessment of Probabilistic Reasoning 65

CHAPTER5 EXPERIMENTAL RESULTS 70
Grouping Data from Model Image and Test Scene 70
Recognition Experiments on 2D Synthetic Images 74
Recognition Experiments with Real 3D Objects 80

Summary 90

CHAPTER6 CONCLUSIONS AND RECOMMENDATIONS 92
Conclusions 92
Limitations of the Algorithm 93
Recommendations 94

APPENDIX 95
Sample Data in Off-line Processing 95
Table of 3D model images for prism 97
Sample CLIPS code 98

BIBLIOGRAPHY 102

VITA 105

LIST OF FIGURES

Figure 1.1 Concept of fuel processing 2
Figure 1.2 Am fabrication 3
Figure 1.3 Manufacturing and inspection of UO_2 tablets 4
Figure 1.4 Robotic assembles 5
Figure 1.5 Pellets buckling 5
Figure 1.6 Indexing as remapping of feature shapes 8

Figure 2.1 Invariant indexing method 15
Figure 2.2 Scheme of the invariant indexing 18
Figure 2.3 Hash table 20

Figure 3.1 Super segment indexing 29
Figure 3.2 Basic representation of model image feature using indexing vectors 32
Figure 3.3 Loop invariance between frame {A} and {B} with respect to frame {U} 34
Figure 3.4 Super perimeter representation 35
Figure 3.5 Stereovision system and triangulation for range determination 38
Figure 3.6 Similarity analyses on loop for 3D object 40
Figure 3.7 3D viewing effects with respect to elevation angle σ changes 43
Figure 3.8 Variation curve showing angle and length ratio 44
Figure 3.9 Topology of feature-based indexing 47

Figure 4.1 Knowledge-based processing in an object recognition system 50
Figure 4.2 Network representation of parallel processing in recognition system 60
Figure 4.3 Four cases of overlapping 61
Figure 4.4 Knowledge based system for three dimensional pattern recognition 69

Figure 5.1 Edge and corner detection from 3D real image 71
Figure 5.2 Computers and sensors used in the experiment 73
Figure 5.3 Model images and test scenes for recognition 74
Figure 5.4 One of eight-test vector sets from test scene A at point *O* 75
Figure 5.5 Results of co-edge search 76
Figure 5.6 Match result after loop search and surface invariance 77
Figure 5.7 Probabilistic reasoning for square object 78
Figure 5.8 Statistics for experiment 1 78
Figure 5.9 Over determined problem in tests scene B 79
Figure 5.10 Super perimeter and surfaces of prism 80
Figure 5.11 Matching results for the same 3D object at varying viewing angles 81
Figure 5.12 One of eleven-test vector sets from test scene C at point *O* 85

Figure 5.13 Result of co-edge search from test scene C 86
Figure 5.14 Final match after loop search and surface invariance 87
Figure 5.15 Recognition results for test scene D 89
Figure 5.16 Final match after loop search and surface invariance 90

Figure A.1.1 Example for super perimeter database for prism model 95
Figure A.1.2 Details of surface invariance between loop 012 and 340 96
Figure A.2.1 List of images for prism database taken at 15-degree interval 97

GLOSSARY OF MAJOR TERMS

Feature is the geometric description of an object, or part of an object, in terms of vertices, edges, loops, and surfaces. A feature characterizes a real object in an image in terms of its contours, similarly to a CAD wire frame.

Model object is a sample object stored as a reference data set for later recognition.

Model image is the set of images that characterize a 3D model object from varying viewing angles. A model image's feature information is stored in the model image database.

Test scene is an image containing unknown object(s), to be identified by pattern matching.

Occlusion denotes a partly visible object due to the presence of another object in the path of view.

Invariance between two images exists if the object's contour is independent of the viewing angle.

Rule is a condition to verify a hypothesis.

Knowledge base (KB) is a collection of formal knowledge employing some formal knowledge representation language. A knowledge base uses rules to verify a hypothesis. A knowledge base forms part of a knowledge-based system (KBS).

Machine Recognition or **Pattern matching** is an algorithm that compares features in a test scene with stored model objects in a database.

ACKNOWLEDGEMENTS

This dissertation owes much to the guidance and support by Dr. Georg Mauer, who has been my faculty adviser. He is an outstanding scholar, and the little I do know regarding the strange process of doing research has been learned in large part from him.

I would like to thank the other members of my supervisory committee: Dr. Meyer, Dr. Trabia, Dr. Wang, and Dr. Yfantis, and Dr. Yim. Their comments were instructive and helpful, and I greatly appreciate their comments and through review of the thesis. Thanks go as well to our external research partners Dr. Bebis and Dr. Li at the University of Nevada Reno, and to Dr. Hwang in Chungnam University for their efforts and insights.

I appreciate the supports of my close friends at UNLV during my five years of study Heesun, Hyekyung, Jaehong, Myunghee, and Yasuo.

I gratefully acknowledge the financial support for my research by the D.O.E and the Office of Naval Research.

I dedicate this dissertation to my parents

Kyesoon and Byungkoo Lee

CHAPTER 1

INTRODUCTION

1.1 Transmutation Fuel Process

The US Department of Energy (DOE) initiated the Transmutation Research Program (TRP) program in 1999. A DOE publication [http://www.ne.doe.gov/aaa/aaa.pdf] describes the program mission as follows: "...the ATW concept transforms plutonium, long-lived actinides, and long-lived fission products contained in spent fuel by changing atomic structures. After transmutation, the new less radioactive isotopes can be stored in a permanent repository." One of the research topics identified by the DOE is "Environmentally Acceptable, and Cost Effective Fuel Processing (Conversion of LWR spent fuel in oxide form to metal fuel for transmutation)."

The large-scale deployment of remote fabrication and refabrication processes will be a requirement for the implementation of transmutation (Meyer, 2001). Fabrication processes for different fuel types differ in terms of equipment types, throughput, and cost. A comprehensive assessment of the issue is presented in NEA, 2000. The scope of the UNLV study on transmuter fuel manufacturing includes a comprehensive study on equipment choices, cost, and plant design, including the spatial simulation of plant operation and of the recovery from unusual events. Because of the stringent requirement

to operate in a hot cell facility, the cost-effective production of transmuter fuel will likely require fuel manufacturing in a largely automated environment.

Examples of comprehensive discussions of transmuter fuel manufacturing issues are found in NEA reports (1999 and 2001) and in a report of the scientific office of the French parliament (1997, in French). The paper by Boidron et al (2000) presents a survey of P&T research efforts. Figure 1.1 illustrates the NEA concept of separating Pu and U from spent fuel and transmuting the minor actinides (MA).

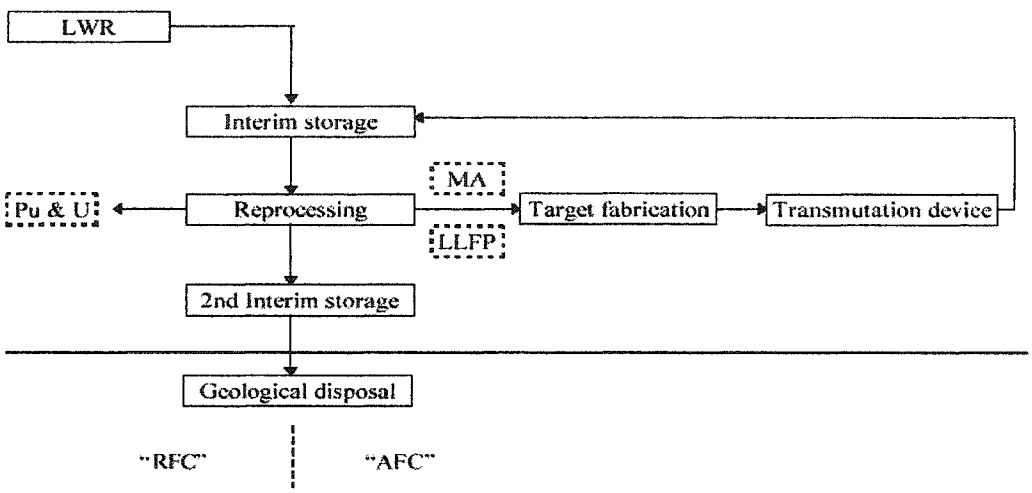


Figure 1.2 **A schematic diagram of back -end of an advanced fuel cycle with minor actinide recycling**

Figure 1.1 Concept for fuel reprocessing (NEA, 1999)

The report to the French senate (1997) estimates the initial costs for a separation plant based on the PUREX process at 5Billion francs or approx. $1Billion, for a throughput of 850 tons of spent fuel annually. Haas et al. (1998) discuss the feasibility of the fabrication of Americium targets in a NEA conference paper using a process developed at the Institute for Transuranic Elements (ITU) in Karlsruhe, Germany. Figure

1.2, quoted from Haas et al. (1998), illustrates the anticipated manufacturing requirements for the fabrication of 1 ton of Am/year. 40,000 Am-pellets would have to be manufactured daily. The author anticipates a need for four pin-filling/welding machines to meet production demands.

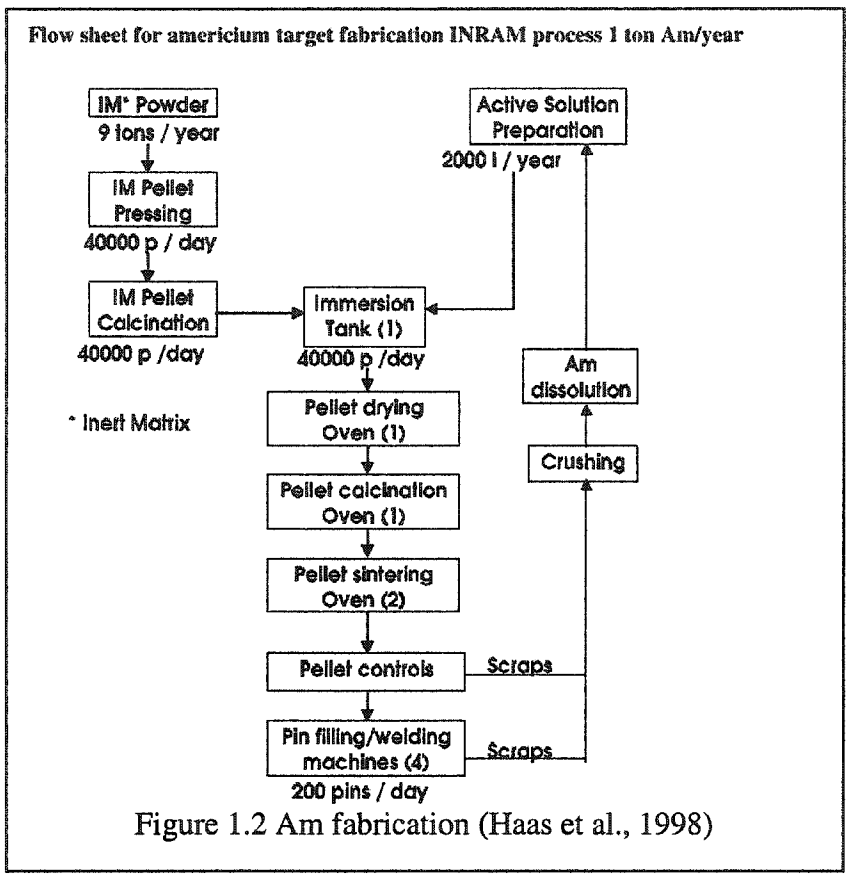


Figure 1.2 Am fabrication (Haas et al., 1998)

The fuel production facility would likely be equipped with the required machinery such as furnaces, blenders, sintering presses, welders, inspection equipment etc.

The material would be transported from one station to the next by robots or by other appropriate forms of automation such as conveyors or part feeders. The design methodology for such manufacturing automation hardware is well understood (Mauer *et al*, 2004). Figure 1.3 shows a schematic of the manufacturing and inspection process for UO_2 pellets and fuel pins at the Framatome Lingen plant in Germany. Framatome's UO_2 fuel manufacturing plant is partly automated. Humans can freely move among the machines and move material as needed. Americium fuel, by contrast, must be

manufactured in a hot cell. For reasons of both cost and production speed, material would be best transported and handled by robots. Since unplanned events can occur, the robots must be flexible to respond to such events, and recover autonomously as often as possible.

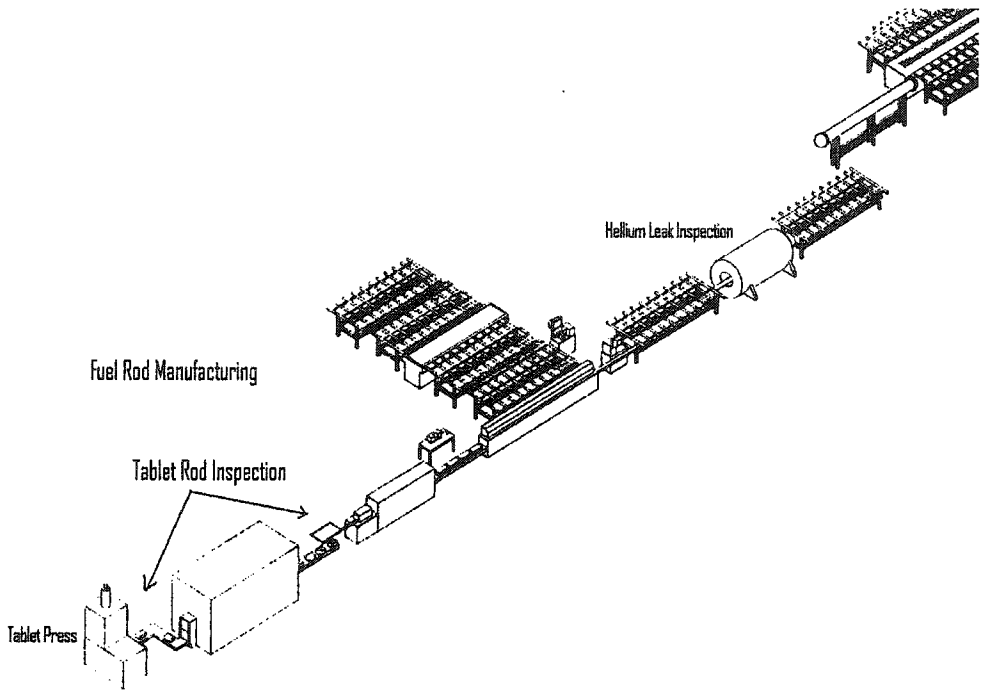


Figure 1.3 Manufacturing and inspection of UO_2 tablets at the Framatome Lingen plant

An essential aspect of remote, automated manufacturing is the real time supervision and control of the process, including process diagnostics and safe recovery from abnormal events. Figure 1.4 illustrates a normal part of fuel manufacturing, the loading of fuel pellets from the pellet press onto a tray fro sintering. An example of an abnormal event is shown in Figure 1.5, where the fuel pins are not properly aligned before insertion into the cladding tube.

The most flexible means for detecting and identifying abnormal events is video monitoring. Cameras can be placed inside and outside the hot cell. Through automated image analysis, all process operations can be monitored, and discrepancies can be detected immediately. In order to perform process monitoring reliably, more research into aspects of image processing such as object segmentation, detection and

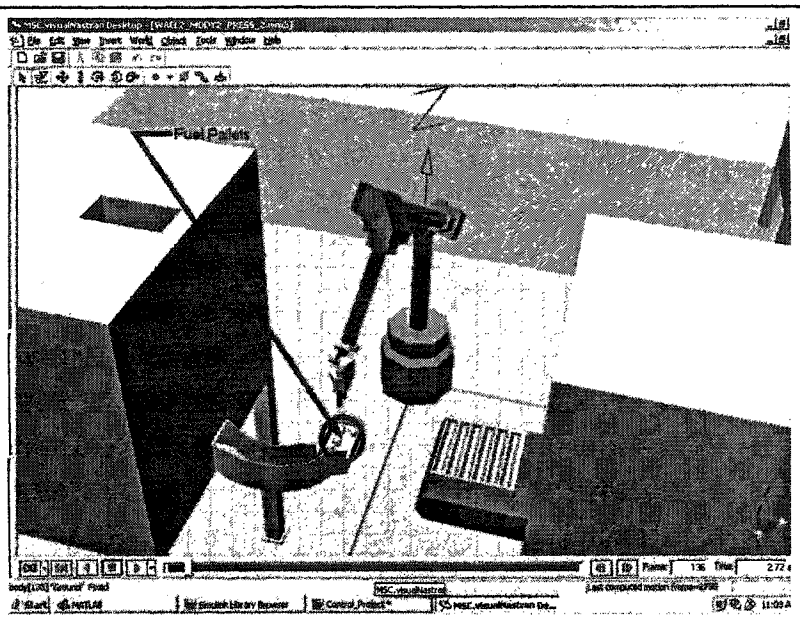


Figure 1.4 Robotic assemblies: removing pellets from the pellet press and loading them onto the tray at right for sintering (simulation)

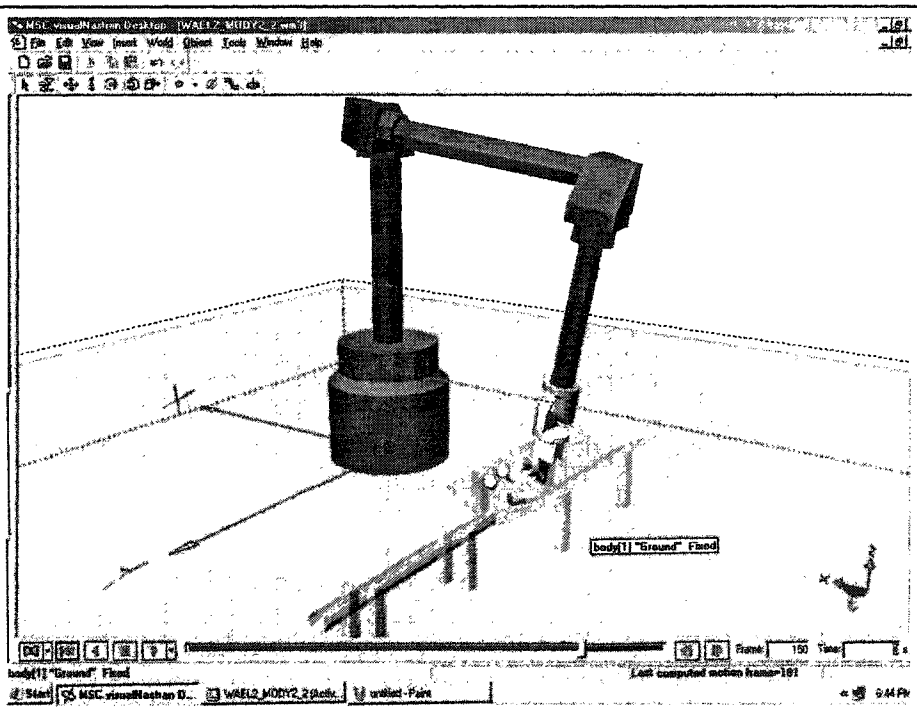


Figure 1.5 Pellets buckling due to misalignment before insertion into the cladding tube (simulation)

recognition is required. In a well-defined manufacturing process, all objects are known, but their appearance to a camera will depend on the spatial location of the camera relative to the object, illumination conditions, shadows, and occlusions, i.e. obstacles in the camera's path of view. Our research focuses on developing algorithms to recognize 3D real objects from any viewing angle even if they are only partially visible.

1.2 Introducing the Indexing Method

Recognizing 3D objects from images has been a challenging task in computer vision. This is mainly because objects may look very different from different viewing positions, and therefore the complexity of the object recognition problem in computer vision lies in the astronomical number of possible combinations of model-to-test feature matches that must be considered. One of the most successful approaches to solve this problem is the model-based object recognition (Chin & Dyer, 1986), where the environment is rather constrained and recognition relies on the existence of a set of predefined model objects. Nevertheless, since there is no prior knowledge of which model points correspond to which test points, recognition can be computationally too expensive, even for a moderate number of models. Therefore various approaches to improve search efficiency have been proposed in the literature.

Indexing is one of several approaches for model-based recognition. In simple terms indexing is a mechanism which, when provided with a key value, is able to rapidly access some associated data. In a book, for example, the index items are topics or key words, the order is alphabetical, and the pointers are to the pages where the items appear. When searching for a key word, it is much faster to use the index than to scan the entire

text for instances of the word. In visual pattern matching, indexing provides a reduction of search time from a linear to a logarithmic scale. In the object recognition process discussed here, image shapes such as edges or surfaces are the keys, and the indexing process recovers model shapes that could have generated them. This is accomplished by a reordering of the data, from their original organization within a 3D model database, to an index ordering which sorts according to feature descriptors for shapes contained with 2D images. Figure 1.6 shows an example: To identify either the house or the camera, we first try to locate unique sets of connected points and match the point sets with the stored point sets in the data base of object models. In the case of Fig. 1.6, those sets of connected points (or loops) at the right hand side of Fig. 1.6 are the key words, and the house and the camera are the pointers. The basic idea of indexing is to find a close match between a stored object model and the test scene, using previously stored *indexing vectors*, which represent segments of the model objects' contours. These indexing vectors are generally obtained by connecting interesting points within the image and contain geometric information describing object features. Non-compatible feature matches are quickly eliminated during the database search. Hence, only the most feasible matches are considered, i.e. those stored model features that could be projected onto, and matched with, the scene features.

Indexing is one of several geometry based object recognition techniques. Another technique is appearance based, where for instance the contour of an object may be stored as a map of pixels. Appearance-based recognition is inadequate in the context of hot cell material handling since it cannot deal with partially occluded objects (Lee & Mauer, 2000). Indexing methods usually employ a hash scheme to efficiently store and retrieve

information about the model features into a *hash table* (Grimmson & Huttenlocher, 1990 & 1992). During preprocessing, groups of feature vectors form descriptions for each object category. These descriptions are the groups of model points and their geometric relations, which are stored in the indexed location. During recognition, groups of connected points from the current image are used to query the hash table.

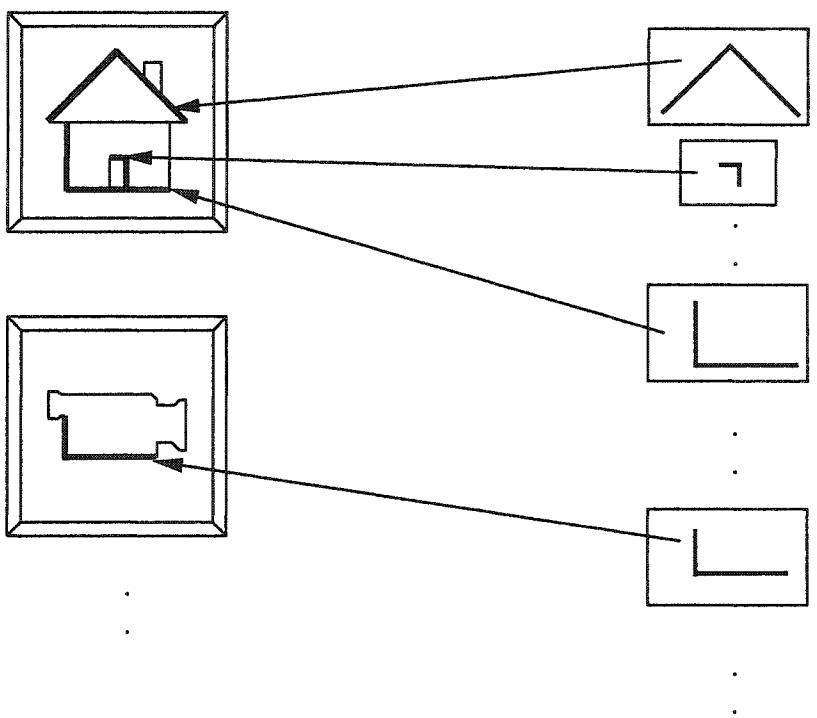


Figure 1.6 Indexing as remapping of feature shapes

Ideally, one would prefer the indexing data computed from a group of model features to remain the same, regardless of changes in the appearance of the model as it is observed from different viewpoints. Such an index is called *invariant.* The main advantage of an invariant index is that a single entry for each group of model features is

sufficient to recognize any test scene, regardless of viewpoint changes. However it is well known that no such invariance exists to represent 3D objects in plane images.

This thesis explicitly addresses the issues of feature variation, ambiguity, and actual versus stored appearance while maintaining the indexing concept. The methodology of indexing primarily concerns accuracy and speed. Speed is the ability of the index to reduce the set of possible matches to a very small number, without eliminating the valid matches. Speed also refers to the lookup time, which must be kept small so as not to offset the reduction in verification due to the index accuracy. Accuracy can be defined as maximum mismatch of Euclidean distance between model dataset and test scene data. In 3D, our algorithm samples the viewing sphere for each 3D model feature and stores the generated set of 2D shape projections in the index. Each sample is represented as a real-valued, three-dimensional feature vector. The data are stored as a tree structure, and we present an approximated nearest neighbor search algorithm that efficiently recovers the data points closest to any runtime shape query.

1.3 Problems in Indexing

In pattern recognition, the primitives are shape and appearance. The shape is defined by geometry such as feature vectors, entry points, and their connectivity. The appearance is pixel based and comprises components such as intensity, surface color, and area moment. Here, indexing vectors describe elements of the model object's shape, and are stored as entries in the model database. With the use of indexing, each entry can have pointers to all occurrences of a feature on all model objects; therefore sequential searches through the set of stored entries can be avoided. While indexing for object recognition

shares the same concepts as other types of indexing, several issues are particular to this application as follows:

Problem 1: *Data are discrete.* Indexing is typically applied to natural features of an image, and features are more naturally represented as continuous real number data than discrete integers. For example, angle data can be represented in the continuous domain $[0, 2\pi]$. Indexing systems must store quantized feature data extracted from an image with finite resolution.

Problem 2: *Shape of feature is ambiguous.* A single set of features in an image may suggest a match with more than one object. Parts of different objects may have identical image appearances. This problem is especially relevant for the 3D-from-2D recognition problem, as a wide variety of 3D shapes can result in the same 2D shape after recognition since shape varies with viewpoint. Occlusion adds to the problem by concealing some portion of a known object.

Problem 3: *Data are ambiguous.* A single image will contain multiple feature sets, so the index must be accessed multiple times no matter whether those queries will lead to correct matches or not. Therefore the same feature may be accessed repeatedly. This is comparable to keywords for the same topic scattered several places in a book index.

Problem 4: *Data are noisy.* Image noise is one of the foremost reasons why the feature-based indexing technique may encounter difficulties. Feature values may be distorted from one image to another due to lighting conditions, digitization, and edge detector properties.

In addition to the problems above, the fundamental properties that determine the performance of an index are the same as for any other computer algorithm: time and

storage efficiency. The trade-offs made by indexing to achieve runtime speedup include not only the time spent building the index, but also the space needed to store it. Since indexing must represent each object as it appears from any point of view, the quantity of stored data can become very large.

1.4 Feature-Based Indexing and Knowledge Base System

The basic indexing method is built for object recognition in 2D invariant. We are using the basic indexing method, however, we gathering those indexing vectors based on shape of feature. The previous indexing method has been built based on point and segment level no matter what the shape looks like, and their matching algorithm is relying upon collecting and compares those indexing vectors as segment on the hash table to verify hypothesis. In preprocessing, our method generates model feature's indexing vectors as a combination of its feature geometry and topology, and in the recognition process, we draw test vectors from any given test scene. Then matching test vectors with model feature's indexing vectors to compare vector magnitudes, inner angles, and transformation invariance of indexing vectors which representing features.

To increase modulus characteristics in the hash table, we employ knowledge-based system on verify match. The basic idea of knowledge-based (KB) system in pattern recognition is to make *rules*, which used to verify hypothesis in every step of recognition as a module so that we can make the recognition process between models and test more compact and fully integrated. Using this artificial intelligence basis on pattern recognition, the complexity of procedure will be enormously decreased when it comes to 3D real image. To achieve this benefit, we must have more localized indexing vectors

and their connectivity of shape features as input model database is essential. In fact, the feature-based indexing method itself is simply represent indexing vectors as topology level geometry. In our research, we construct feature of geometry using its points and segments vectors as well as their form of transformation relations to identify model object in a test scene until topology level. To construct model image's database as tree structure of feature-base we classify each surface of model image feature as a set of surfaces, we classify each surface and feature of model image as a set of junctions and loop, and we classify each junction and loop of model image as a set of edges and vertex. During the recognition, every test vector from test scene compared with these edges and loops in the model data. And its transformation (translation and rotation) invariance examined as well to find match any surface in the tests scene.

The major aspects that we should note are as follows:

1. **Invariant indexing**

It is well known there are no general invariants under three-dimensional perspective projections, but we have chosen to store similarity distributions for two-dimensional patterns instead. However during the 3D recognition itself, the algorithm execute the matching process as if there is invariance between model image and test scene. And the difference is cumulated as error.

2. **Generality**

In this thesis, the database of model objects was limited to objects with clearly defined contours. All groups of objects used for data base construction are represented in the database as combinations of line segments. In addition to contours, the potential exists to use curved edges, texture patches, and color regions as features. Some ideas for

integrating color with the current set of geometric features and curved edges are presented in the section on the future work.

3. **Similarity: interpolation between two similar stored views**

Although there is no general invariance in 3D, similarities exist as long as the changes of viewing angle are small. We drive probabilistic function to estimate the boundary of similarity of pose and interpolate their approximated view between two similar 2D views. Such probabilistic viewing effect has been proven method for dealing with small change of viewing angle.

4. **Image noise**

Image noise induces broken line segments of contours. When creating the database, noise effects can be corrected manually by restoring the actual start and end points of every line feature.

1.5 Thesis Organization

Chapter two explains the details of invariant indexing method and introduces previous related work.

Chapter three presents the feature-based indexing method and a probabilistic interpolation method for estimating the pose of objects.

Chapter four introduces a knowledge-base system for recognition and probabilistic reasoning.

Chapter five shows implementations of our algorithm on 2D synthetic images as well as 3D real images, including the identification of partially occluded objects.

Chapter six presents conclusions and outlines possible future work.

CHAPTER 2

INVARIANT INDEXING METHOD AND PREVIOUS WORK

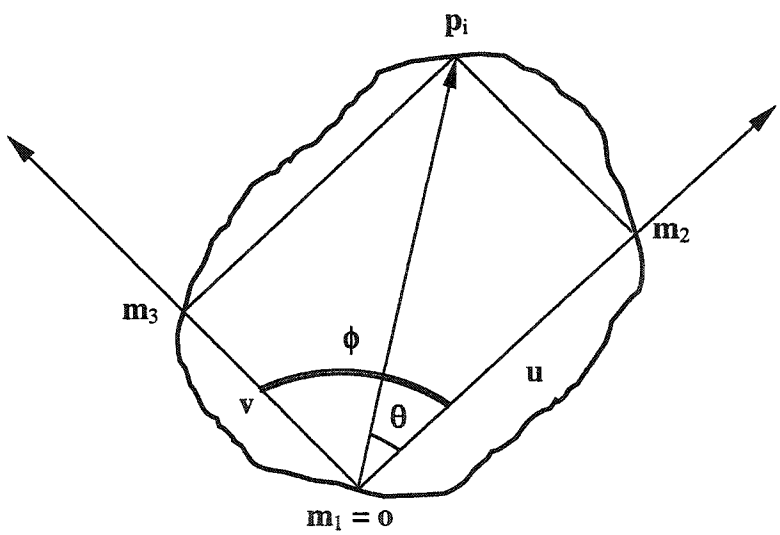
In this chapter, we introduce the methodology of the invariant indexing method and discuss related work. We also demonstrate hash table design and analyze its complexity. Lastly, we discuss the assumptions applied to finding the optimal match between stored indexing vectors and those of test scene.

2.1 The Invariant Indexing Method

The basic idea of geometry-based matching using indexing is summarized by Grimmson & Huttenlocher (1990) as follows: "Objects are represented by collections of features, or interest points. A match of a model image to a test scene consists of a subset of the model features being mapped onto a corresponding part of test scene using a certain transformation of some type." Geometric hashing has been used with various types of transformation, including 2D and 3D rigid body motions and affine approximations to orthographic projection. Here, we consider 2D affine transformations (i.e., each image point $\mathbf{x}'$ is obtained from a model point $\mathbf{x}$ by $\mathbf{x}' = \mathbf{A}\mathbf{x} + \mathbf{b}$ where $\mathbf{A}$ is a non-singular 2×2 matrix, and $\mathbf{b}$ is a 2x1 column vector). Similar analyses apply to the other types of transformations.

The idea behind the hash table is that each k-tuple of model points forms a basis frame for a coordinate system that is invariant under possible model transformations. Thus all model points can be stored in a transformation-invariant manner by rewriting them in terms of this reference frame.

The rewritten model points serve as indices into the hash table, where the corresponding basic k-tuples are stored. The number of points in a basis, k, depends on the particular type of coordinate transformation. Here, we consider 2D affine transformations, so $k = 3$ (see Figure 2.1).



$\mathbf{p}_i - \mathbf{o} = \mathbf{u} + \mathbf{v}$; for any point $\mathbf{p}_i$ on the contour of a image

Figure 2.1 Invariant Indexing Method
3 points defining the basis frame ($\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{m}_3$); $k = 3$

We first describe the method without the presence of sensor or numerical round off errors. Indexing data for each model are entered into the hash table as follows:

(1) Choose an ordered set of three non-collinear points from model image $\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{m}_3$ as a basis, formed by an origin $\mathbf{o} = \mathbf{m}_1$ and a pair of axes $\mathbf{u} = \mathbf{m}_2 - \mathbf{m}_1$, $\mathbf{v} = \mathbf{m}_3 - \mathbf{m}_1$.

(2) For each additional model point $\mathbf{p}_i$, rewrite the coordinates of $\mathbf{p}_i - \mathbf{o}$ in the affine basis defined by the axes $\mathbf{u}$, $\mathbf{v}$, i.e., find α, β such that $\mathbf{p}_i - \mathbf{o} = \alpha\mathbf{u} + \beta\mathbf{v}$.

(3) Hash into a table using the indices (α, β), and store at that point in the table the basis triplet $(\mathbf{o}, \mathbf{u}, \mathbf{v})$.

(4) Repeat this process for all possible choices of model bases (i.e., for all ordered triplets model points). This results in a table indexed by affine-invariant coordinates. Any pair of (α, β) values can be used to retrieve the model bases (if any) for which some model point $\mathbf{p}_i$, has the affine-invariant coordinates (α, β).

We begin with an expression for α, β in the no-noise case. Without loss of generality, we set the origin $\mathbf{o}$ to zero. By Figure 2.1, we have

$$\alpha = \frac{\langle \mathbf{p}, \mathbf{v}^{\perp} \rangle}{\langle \mathbf{u}, \mathbf{v}^{\perp} \rangle} = \frac{p \sin(\phi - \theta)}{u \sin(\phi)} \qquad (2.1)$$

$$\beta = \frac{\langle \mathbf{p}, \mathbf{u}^{\perp} \rangle}{\langle \mathbf{v}, \mathbf{u}^{\perp} \rangle} = \frac{p \sin(\theta)}{v \sin(\phi)} \qquad (2.2)$$

At recognition time, the hash table is used to determine which models are present in the test scene. A set of triple points from the test scene that corresponds to the model image will yield coordinates that coincide with entries in the hash table (because the model is stored in the table in terms of all possible bases, and because the representation is invariant under any affine transformation). If we have selected a model image database that has right corresponds to the test scene, then all other points from there will yield (α, β) pairs that match with the model basis in the hash table.

Therefore the procedure at recognition time is:

(1) Choose a set of three points from the test scene s_1, s_2, s_3 to form a basis, formed by an origin $\mathbf{O} = s_1$ and a pair of axes $\mathbf{U} = s_2 - s_1$, and

$$\mathbf{V} = s_3 - s_1.$$

(2) For each additional sensor point $\mathbf{p'}_i$, rewrite the coordinates of $\mathbf{p'}_i - \mathbf{O}$ in the affined basis defined by **U**, **V**, i.e., find α',β' such that

$$\mathbf{p'}_i - \mathbf{O} = \alpha'\mathbf{U} + \beta'\mathbf{V}.$$

(3) Index into the hash table using (α', β'), and retrieve the stored set of bases, each of which is a possible candidate match. Each time a given basis is retrieved from the table a histogram counter for that basis is incremented. Repeat for all additional sensor points.

(4) Once all sensor points have been hashed, the histogram contains votes for those model bases that could correspond to the current sensor basis, (**O**, **U**, **V**). If the peak in the histogram for a given model basis, (**o**, **u**, **v**), is sufficiently high, then this basis is selected as a possible match.

The transformation from model to image coordinates can be computed from the corresponding model and image bases.

(5) The entire operation is repeated for all possible bases (i.e., all image point triples) until a match is found. The histograms are reset at each iteration.

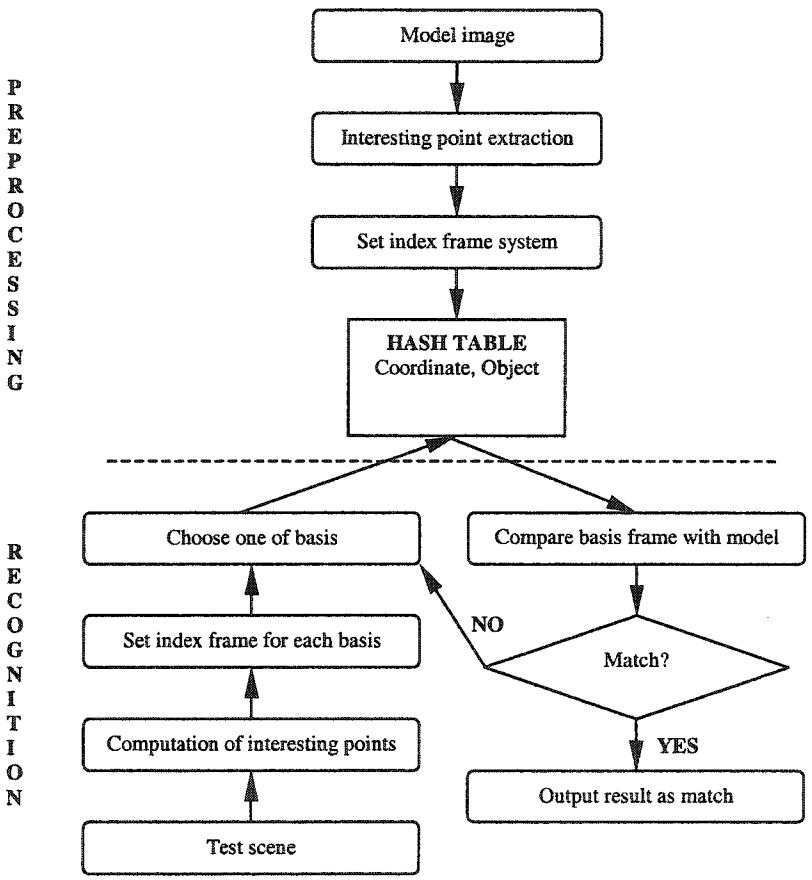


Figure 2.2 Scheme of the invariant indexing

Clearly, the effectiveness of the geometric hashing method depends on the distribution of the entries in the model hash table. In the worst case all model features hash to the same location, therefore all model bases would receive a vote. The hash tables must therefore contain an appropriate amount of detail, but the analysis implies that this can often not be ensured a priori. In particular, the presence of signal noise results in a range of values for α and β rather than a single value for a point, making it difficult to form appropriately sized tables.

As shown in Figure 2.2, the algorithm consists two major parts. The first one is preprocessing, which is applied to the model points. This step does not use any information about the scene and is executed off line before actual matching is attempted. The second part uses the data from preprocessing to match the models against objects contained in the scene. The time to perform this second step is the actual recognition time.

To analyze geometric hashing, we use some formal tools that have been applied to other recognition methods. Two questions arise: First, how does measurement noise affect the parameters used in recognition? We compute the estimate for the range of affine coordinates of a point consistent with bounded amounts of sensor error. Secondly, given that each feature consistent with a range of entries in the hash table, how does the probability of a false positive identification of a coordinate basis change with increasing amounts of error, occlusion, spurious data, and numbers of model images? We use a statistical model to derive estimates for the probability of a false positive, as a function of error and numbers of sensor and model images.

2.2 Building a Hash Table and Complexity Analysis

For the sake of clarity, we describe our algorithm for the simple case where the model image database contains only one object. However, the presentation applies also to the general case, where a number of models may appear in the scene. As mentioned previously, the models and objects in the scene are described by sets of interest points.

Given m points on a model image and n points in the test scene, there are $O(n^m)$ ways to match the model image points to the test scene points. Since such an exponential complexity is unacceptable for object recognition algorithms, various attempts were made to prune the space of possible matches. Some of them try to employ efficient tree search techniques, where the pruning is based on geometric constraints. However, the search still remains exponential in the number of scene features for recognition of partially occluded objects. Figure 2.3 shows a typical hash table format in pattern recognition between a model object with m points and a test scene with n points.

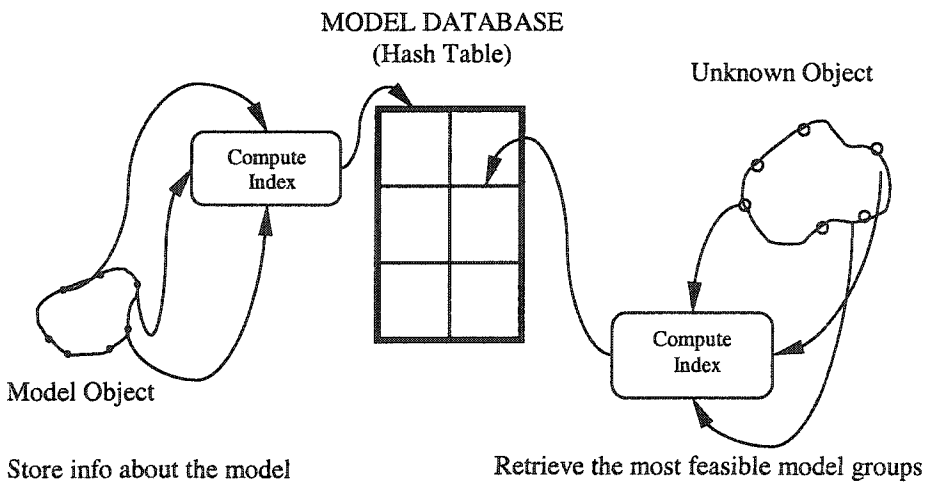


Figure 2.3 Hash table

To overcome this exponential complexity, one may observe that the transformation of a rigid object is completely defined by the transformation of a small number of the object's points. This geometric observation is at the core of the so-called indexing and alignment techniques. For example, it is well known that an affine transformation of a plane is uniquely defined by the transformation of three ordered non-collinear points. Moreover, there is a unique affine transformation that maps any ordered non-collinear triplet in the plane to another ordered non-collinear triplet. Hence, we may extract interest points on the model image and the test scene and try to match non-collinear triplets of such points to obtain candidate affine transformations. Matching the transformed model against the scene can verify each such transformation.

However, the complexity of such a scheme is rather unfavorable. Given m points in the model image and n points in the test scene, the worst case complexity is $(m \times n)^3 \times t$, where t is the complexity of matching the model against the scene. If we assume that m and n are of the same magnitude and t is at least of magnitude m, the worst-case complexity is of order n^7. One way to reduce this complexity is to classify the points in a distinctive way so that each triplet can match only a small number of other triplets.

Assume that we are given an image of a model where m interest points have been extracted. For each ordered non-collinear triplet of model image points, the coordinates of all other m-3 model image points are computed taking this triplet as an affine basis of the 3D plane. Each such coordinate (after a proper alignment) is used as an entry to a hash table, where we record the basis-triplet at which the coordinate was obtained, as was the model (in case of more than one model). This encoding of each point in all possible affine basis coordinates gives us an affine invariant representation of the m point set,

which will enable efficient indexing in the recognition stage. The complexity of the evaluation step is of order m^{m-3} per model. New models added to the database can be processed independently without re-computing the hash table.

In the verification (recognition) stage, we consider the image of a test scene from which n interest points have been extracted. We choose an arbitrary ordered triplet in the test scene and compute the coordinates of the scene points, taking this triplet as an affine basis. For each such coordinate, we verify the appropriate entry in the hash table, and for every pair (model, basis triplet) that appears in the hash table, we vote for the model and the basis triplet as corresponding to the triplet chosen in the scene. (If there is only one model, we have to vote for the basis triplet).

If a certain pair of triplets (model, basis triplet) scores a large number of votes, we decide that this triplet corresponds to the one chosen in the scene. The uniquely defined affine transformation between these triplets is assumed to be the transformation between the model image and the test scene. This candidate transformation is then verified in two successive steps, first by performing a nearest neighbor match of all candidate model image points, and then by direct verification of all model edges under the appropriate transformation versus the test vectors (this time considering all edges, not only interest points). If the current triplet does not score high enough, we examine another basis triplet in the test scene.

For affine transformations, a basis frame can be created from a set of three points of interest. Within this frame, the coordinates of all other points co-planar with the original three points are invariant. During recognition, any three-interest points from a test scene are chosen as a basis, and the remaining points are hashed into the hash table.

If a match with the model image frame exists the score is recorded. The efficiency of the method is only possible when each of the model databases is redundantly stored in the hash table at the cost of increased memory requirements.

2.3 Related Work

The process of identifying indexing match 2D test scenes using 2D model data is relatively straightforward since feature properties like angles, edges, and invariant to 2D rotation (with respect to z-axis), translation and scaling. Stein & Medioni (1992a) use larger feature groupings, which they call "super segments", to compute their index. These are chains of connected straight-line segments, generated by polygonal approximations to image curves. The complete set is used during storage and retrieval, in order to mitigate the effects of the distribution caused by approximating curves with straight lines. The multi-dimensional index vectors in this method consist of the angles between consecutive segments plus the "eccentricity" of the entire group, which is the second moment of inertia of the vertices of the super segment. Our method also uses segment chains as one type of grouping to generate index vectors. Instead of "super segments", however, we use groups of non-collinear points set to find a match.

For 3D object recognition, Stein & Medioni (1992b) extend their 2D work to 3D range data. The analysis is based on two types of features: super-segments, generalized to 3D, and, for surfaces that do not have good polyhedral segmentations, a novel shape descriptor for local surface patches, which the authors call "splash".

While no invariants exist for general 3D feature sets projected onto a single 2D image (Clemens & Jacobs, 1991), there has been considerable progress in the area of

constrained 3D feature sets. These are special configurations of shapes/features from which invariant features may be computed.

For example, Rothwell, Zisserman, Mundy & Forsyth, (1992) derive invariants from concavities in planar curves. Four special points can be defined for any concavity; these points are used to set up a canonical reference frame. An image curve is mapped into this frame, where it takes on a unique, invariant shape. This type of invariant is useful if the objects in the database contain the required concave curve features. However, the set of objects that can be indexed is quite limited.

Another approach by Weinshall, (1993) and Weiss & Ray, (2001) requires multiple images. The invariants are matrices whose elements are functions of an ordered set of point coordinates. This type of matrix equation will have an exact solution if the input point set has same correspondences to each other in orderly manner.

Others have attempted to recognize objects by building and then solving matrix equations (Ullman & Basri, 1991), (Bebis, Georgiopoulos, Shah & Vitoria, 1998). The matrix equations become over-determined problems in most general cases, and can be solved by singular value decomposition (SVD). An exact match is found by optimizing the coefficient values.

Geometric hashing (Landam, Schwartz & Wolfson, 1990) is closely associated with 3D from 2D method. In these authors' method, features such as corners and curvature extremes are characterized as "interesting points" and used to describe shapes.

2.4 Limitations of the Indexing Method

Currently, there is no widely accepted indexing solution for object recognition. Current methods are still exploratory, and therefore the engineering tolerance exists in the solution. The absence of a single preferred method is also due to certain practical difficulties regarding:

- Acquisition of the object models;
- A number of Number of parameters that affect performance;
- Large range of variability of many parameters.

Object model generation is problematic due to the time required to accurately measure the objects manually. The tediousness of this process is one reason why most databases contain only small numbers of models.

Problems that complicate the indexing algorithm include:

(1) Model representation: Approaches vary according to the choice of features, image data dimensionality (2D or 3D), and model feature dimensionality (2D or 3D). A method may exhibit very poor performance when applied to an object class favored by a different method.

(2) Lighting condition: Indexing and recognition are directly affected by the location and intensity of light sources. Lighting influences on feature detection by through spectrum as well as varying levels of shadows and contrasts. The range of possible lighting conditions is extremely large.

(3) Object occlusion: This variable has never been qualified for use in recognition experiments. One definition might be "the percentage of the image area of the target object that should be visible but is concealed". Unfortunately, the degree of

occlusion can only be determined post-recognition, using the back-projected object, thus it cannot be measured if indexing fails. To date, more qualitative, descriptive measures have been used, for example "significant" or "mild" occlusion.

(4) Ratio of object size to image size: In general, the larger the object appears in the image, the easier it is to recognize, because feature detectors give better results during both preprocessing and recognition. A further consideration is the pose of the object with respect to the camera, since some viewpoints give rise to a greater number of features, or a more useful set of features, than others.

(5) Degree of clustering: A large number of distracters in an image not only slows indexing down, but increases segmentation and grouping errors, which influences the indexing.

Because of these difficulties, researchers commonly provide selected example images on which their algorithms have been successful, rather than providing a thorough coverage of all imaging situations. Typical assumptions are (i) that the object occupies a large fraction of the scene (which means that the view is close to orthographic projection); and (ii) that lighting conditions are not extreme (implying that a large fraction of the light is ambient, non-directional light). These assumptions are currently acceptable because the problems of occlusion, clustering, and large database size are so difficult that concerns about stark contrasts are of relatively minor importance. A demonstration of fast and accurate recognition under these two assumptions, for a large database in real, clustering images, would be considered a very strong result.

The scope of this thesis is bound by two restrictions: First, we will only consider objects with straight edges. This limitation has several advantages: matching between 3D model segments and 2D image segments is straightforward; grouping relationships between segments are relatively easy to specify; and there exists a rich set of real-valued features that can be derived from sets of straight edges (such as various angles and edge length ratios). The disadvantage is that the class of objects that can be modeled is restricted to non-convex polyhedral.

The second restriction is to consider only rigid objects. This means that we cannot claim to recognize generic objects, such as "any square", but any specific objects, (e.g. "square with length of L"). Generic objects are those defined by their component parts, and relationships between their components. Parameters, which affect the specific object shape, are allowed to vary. In a word, cannot classify similar objects without detailed extra descriptions.

Given these restrictions, it will be demonstrated that the approach to indexing proposed in this thesis is a strong competition with other methods, while solving a more difficult problem than most previous attempts. Our experimental approach will be to explore the properties of our technique using 2D synthetic images first where its ground truth is concrete, then we apply into 3D real objects.

CHAPTER 3

PREPROCESSING USING FEATURE-BASED MODELING

This chapter describes methods for representing and classifying the topology of objects by using indexing vectors. To represent features, we define the terms 'loop' and 'surface' in the context of indexing. The indexing vectors will be described as attributes of model features. Even though the discussion focuses mainly on methods to build models, it will also extend to implications for the recognition process.

3.1 Previous Work on Feature-Based Indexing

The literature contains many attempts for shape-based recognition using an invariant indexing method. However, only a few authors are dealing with recognition using 2D images. Several similar concepts exist for feature-based recognition (Beis & Lowe, 1999) and we introduce here two methods similar to ours. One is the super segmentation method and the other one is using four straight segmentation chains.

3.1.1 Super Segmentation

Medioni & Stein (1992a) developed super segmentation. Figure 3.1 shows this method: it uses indexing vectors and their cardinality, arc length, angles, locations, orientation and eccentricity to represent feature data. Distance, angle, and direction become the constraints for recognition.

For a 2D object, it is natural to use boundaries as their representation since there is no change of view with z-axis rotation. The super segment representation of a model or a scene is based on polygonal approximations. The polygonal approximation of a curve captures some of the curvature information in the form of the angle between consecutive segments. Several segment links then represent the entire 2D shape. During model generation, we collect those chain segments and store them with prior knowledge of their super segmentation information. During recognition we align each of these segments with the test scene, starting from the lowest link of the chain to the highest, until the super segment matches.

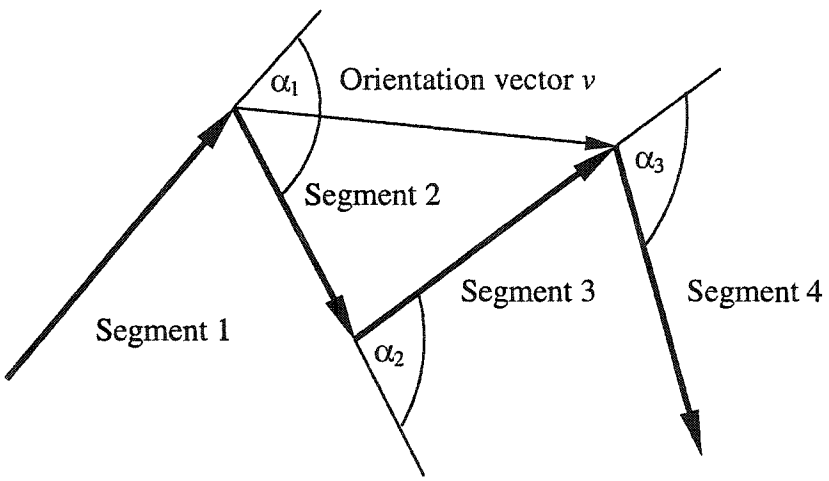


Figure 3.1 Super segment indexing

3.1.2 Four-Segmentation Chain

Beis & Lowe (1999) developed a much simpler algorithm from the super segmentation method, creating a significant improvement to invariant indexing. Their idea allows the

reduction of the number of iterations and the memory size for recognition. The methodology consists of dividing all boundaries of the model shape into the links of four straight lines as segment chains. If a match with a test scene point set exists, the error is calculated and stored in the *bin*. After all the comparison between model and test scene is completed, the best bin will be selected as one of the most possible match. The match can be conducted by three or five-segmentation chains or even more, and for the n segment for n test points it will be the same as super segmentation mentioned above. Therefore this method can be considered as a simplified version of the super segmentation method. Resulting from the simplification, the accuracy of matching results decreases. The authors improve recognition reliability again by employing nearest neighbor search (NN search) and best bin first (or BBF) techniques.

3.1.3 Our Method: Feature-Based Indexing

Our method was developed from the concept of Beis & Lowe (1999). Instead of using multiple-segment chains, we employ three non-collinear point sets or *loops* for matching. We term this method *feature-based indexing*. To improve the accuracy and efficiency of the recognition process, we develop a rule-based algorithm and define quantitative criteria for determining the *degree of uncertainty*. Like the segment-chain method described in the previous section, our algorithm seeks to identify shapes in the test scene directly, regardless of whether the object is plane or three-dimensional. Table 3.1 shows a comparison of the indexing methods discussed here. The details of the recognition algorithm such as rule-based search and degree of uncertainty will be introduced in the next chapter. In this chapter, a discussion of the process for defining and building model database using feature-based concept.

Table 3.1 Recognition algorithm comparison of current indexing	
Method (year)	**Object Types: Acq. /Recog.**
Invariant Indexing (1987)	2D/2D only, using Invariant frames
Super segment (1992)	3D/3D needs range data
Four-segment chain (1999)	Range data needed 3D from 2D: Multiple-segments chains
Our method	3D from uniform scaled multiple 2D image: Loops

3.2 Definitions of Attributes in Feature-Based Modeling

We define attributes of feature-based indexing for shape descriptions of model objects as follows:

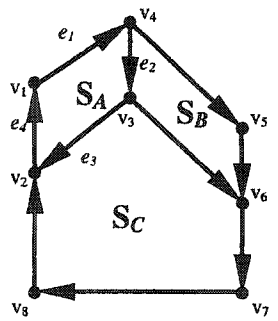
(1) *Vertex*

A *vertex* or *interesting point* (v_1, v_2...) is defined as a point's set of (x, y) coordinates. See Figure 3.2 for examples. Vertex coordinates result typically from a corner detection algorithm applied to contour(s) within the image (Canny 1986, Medioni & Yasumoto 1987). Those corners are usually transition points between convex and concave curve segments. During model generation, the input sequence of vertex coordinates is organized in a pre-determined and orderly manner. However, during recognition, the point order of sequence is random.

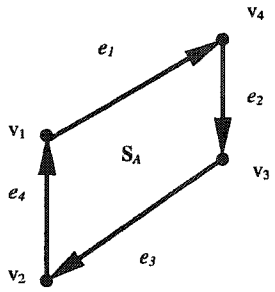
(2) *Edge*

An *edge* (e_1, e_2...) is defined as a vector from one vertex point to another. The set of edges describes the contours of the object. The arrows in Figure 3.2 (b) represent the edges that connect vertices along the contours of the object in Figure 3.2 (a). As a vector, an edge contains information on magnitude and direction.

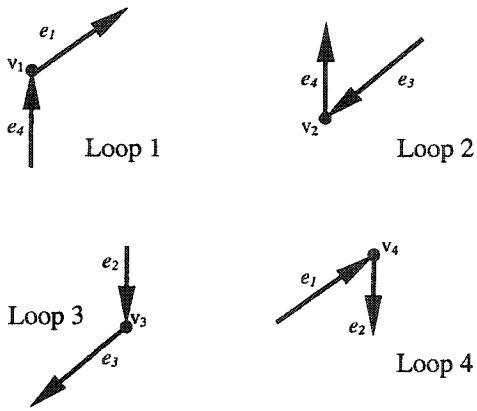
(a) Object Original Image



(b) Representing real image features by edge (contour) vectors



(c) Representing surface A (S_A) in terms of edge and vertex



(d) Four loops consist surface A (S_A)

Figure 3.2 Basic representation of model image feature using indexing vectors

During model generation the edge data are entered into the model database using prior knowledge. During the recognition process arbitrary test vectors connect every set of two points. A simple recognition algorithm would have to compare each test vector with all model image edges.

The edges become elements of indexing vectors that represent the shape of model objects. Edges also become elements of loops and surfaces, to be introduced below. In

3D, the directionality of edges plays a key role in identifying surface invariance. In case of n vertices, the order of edges is $O(n)$.

(3) *Test Vector and Co-Edge*

In the test scene (during recognition), we introduce *test vectors* and *co-edges*. A test vector connects any set of two points. We compare the magnitude of edges in the model database with test vectors from the test scene. Edges in the model database represent the shapes of surfaces. By matching the magnitudes of test scene edges with those from the model image database, we retain only the correct test vectors. These correct vectors are classified as *co-edges*. For n vertices in the test scene, the order of test vectors is $O(n^2)$.

(4) *Loop and Junction*

A *loop* is defined as a non-collinear triplet of vertices in the model image. As seen in the examples of Fig. 3.2(d), loops assume triangular shapes. A loop is the most fundamental element in both the preprocessing and recognition processes. Even after co-edges have been collected from test vectors, false positive indexing vectors still exist. We perform an inner angle match of loops in the model database with those of the test scene. No-matching co-edges from the previous level of recognition are eliminated.

A *junction* is defined as a loop that has more than three edges from its middle point. The junction can be described as multiple combinations of loops.

(5) *Surface*

A *Surface* is defined as a set of loops that represents plane 2D geometry. To describe a surface contour using loop connectivity, we use the *transform invariance* property of the loops, which stems from the fact that all vectors on the same surface are coplanar. Figure 3.3 illustrates transform invariance: two adjacent loops A and B are linked by the

same invariant vector v with fixed length and orientation. Equations (3.1) and (3.2) describe the invariant transformation of loops A and B to the same fixed base {U}, see dashed lines in Fig. 3.3.

$$\begin{bmatrix} {}^{U}Loop \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{U}_{A}R & {}^{U}Loop_{AORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{A}Loop \\ 1 \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} {}^{U}Loop \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{U}_{B}R & {}^{U}Loop_{BORG} \\ 0 \quad 0 \quad 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{B}Loop \\ 1 \end{bmatrix} \tag{3.2}$$

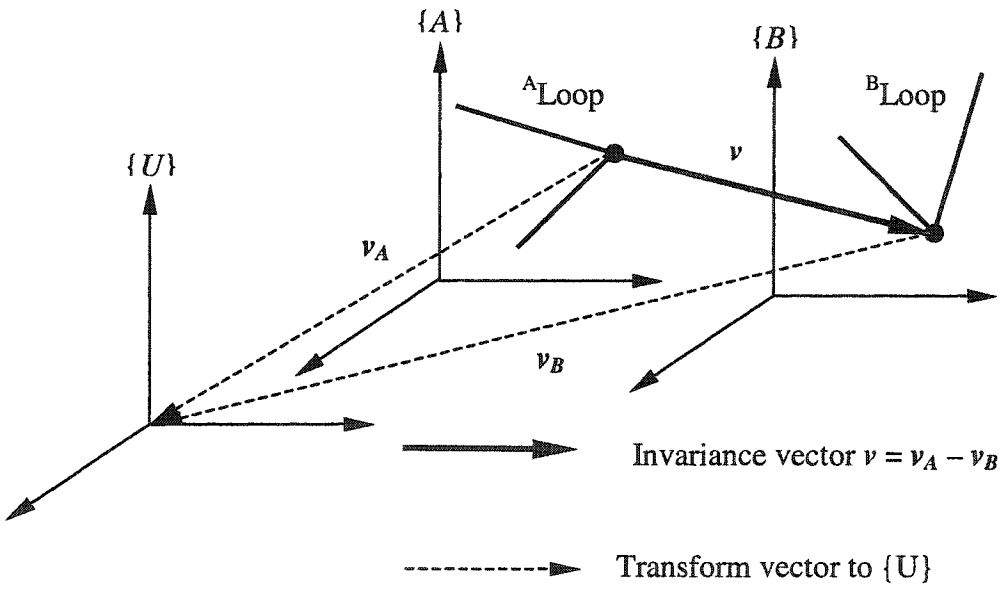


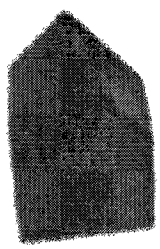
Figure 3.3 Loop invariance between frame {A} and {B} with respect to frame {U}

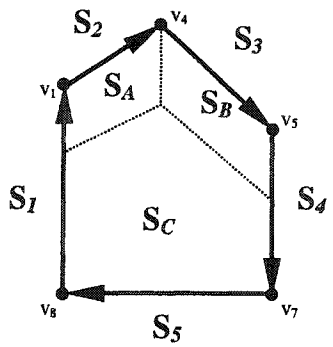
Clearly, the vector $v = v_A - v_B$, describing the distance between A and B, must also be invariant. During model generation, we create initial loop pairings for surface

points in the model image. Other points belong to the same surface if they map to frame {U} with the same transformation. During recognition, the same procedure tests for transformation invariance of loop pairings between model image and test scene. Due to image noise and discretization, the match is not normally perfect. The invariance estimate therefore includes an estimation of its uncertainty derived from the invariance of the paired loops from model image and test scene. Potential loop pairings are evaluated using their transformation, uncertainty, and topological relations between features, resulting in the adoption of the least ambiguous pairings.

(6) *Feature and Super Perimeter*

A *feature* is defined as a set of surfaces in terms of junction connectivity and their neighborhood information. In the case of a 2D image, the feature is a single surface. Figure 3.4 shows an instance of a feature with a total of four surfaces. Each surface is connected to its neighboring three surfaces S_A, S_B, S_C, respectively, and to its *super perimeter* vectors, S_1, S_2, S_3, S_4, and S_5 (bold arrows).





(a) Original 3D real image

(b) Super perimeter representation (Bold arrows)

Figure 3.4 Super Perimeter Representation

The *super perimeter* is defined as a connection of loops surrounding all contours of a feature. The relation of the super perimeter to the individual surfaces is expressed as a sequence of connected surfaces, where each surface S_i is expressed as a sequence of loops in the connectivity table (see Table 3.2). The connectivity table format is: <<connection-type, junction-type 1>>. Here junction 1 is branch of three dotted line in Figure 3.4 (b). Surface S_A by itself is expressed as << S_A; L_1, L_2, L_3, L_4 >> from Figure 3.2, and the super perimeter is expressed as << Super; S_1, S_2, S_3, S_4, S_5 >>. When describing a planar surface, the relations between junctions become invariance of loops: <<connection-type, loop-type 1, loop-type 2>>.

Defining the super perimeter of a 3D feature creates a convenient format for describing the geometrical relationships between the plane surfaces projected from a 3D shape onto a 2D image. Figure 3.4(b) shows the super perimeter representation for the 3D feature seen in Figure 3.4(a).

Table 3.2: Connectivity Table for the Surfaces in Fig. 3.4

Surface	Connectivity	Number of Loops
Super perimeter	Super; S_1, S_2, S_3, S_4, S_5	Five
Surface A	S_A; L_1, L_2, L_3, L_4	Four
Surface B	S_B; L_5, L_6, L_7, L_8	Four
Surface C	S_C; L_9, L_{10}, L_{11}, L_{12}, L_{13}	Five

The concept for recognizing 3D features using the super perimeter and 2D surface data originated from similarity analysis conducted by Ben-Ari (1990), Burns *et al* (1993),

and Olson (1995). Their similarity analyses will be introduced in the next section. The procedure of model generation for 3D features is the same as building surface databases in 2D. During the recognition, whenever the super perimeter is recovered we can quickly establish feature connectivity using candidate data sets from the model database.

(7) *Curved Objects*

Though our recognition algorithm does not primarily concern curved edges, we briefly remark how to build curved edges. Usually curved object suggests the use of sharp convexities, deep concavities, and zero curvature points. And to represent their curved edges, one must include one more points to interpolate their curve equation. The reason we ignore the curved edge is it can be defined as multiple connection of straight line if we try to find individual pixels along the curved contours. And for the same reason it is also increasing the size and complexity of model image database to handle them. By doing so, such curved edges can be represented as sharp-edged polyhedral geometry without any curves. Therefore, in this thesis, we implement our algorithm only focus on polyhedral geometry.

(8) *Dimensional Inspection*

Camera images are projections of 3D scenes onto a plane. A single plane image does not contain range information about the distance between the camera and an object in the scene. Ranges data can be derived, however, from additional information, such as triangulation from stereovision, see Figure 3.5. Knowing the range permits the measurement of dimensional properties of objects in the scene. Referring to Figure 3.5, we assume that the distance between camera d and the focal length f of each camera are known. The *disparity* between two image points x_1 and x_2 is inversely proportional to the

range (or depth) r. For the stereo system in Figure 3.5, the approximate range of point P results as:

$$Range = r = \frac{d * \sqrt{f^2 + x_1^2 + x_2^2}}{|x_1 - x_2|} \qquad (3.3)$$

where

r range or depth

d distance between the lens centers (camera baseline)

f focal length of the lens

x_1 distance of point from center of image 1

x_2 distance of point from center of image 2

$|x_1 - x_2|$ disparity

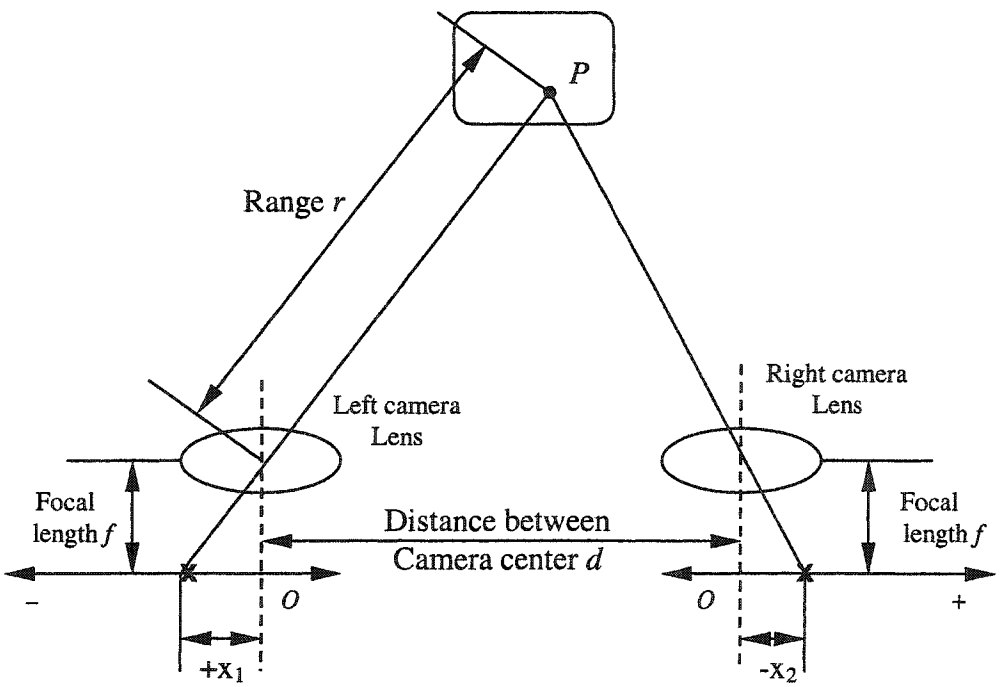


Figure 3.5 Stereo vision system and triangulation for range determination

In remote manufacturing, knowing the range is essential for the visual inspection of manufactured parts. Besides, stereovision, the range can be determined by other sensors (e.g. ultrasound or triangulation).

3.3 Probabilistic Similarity Analysis

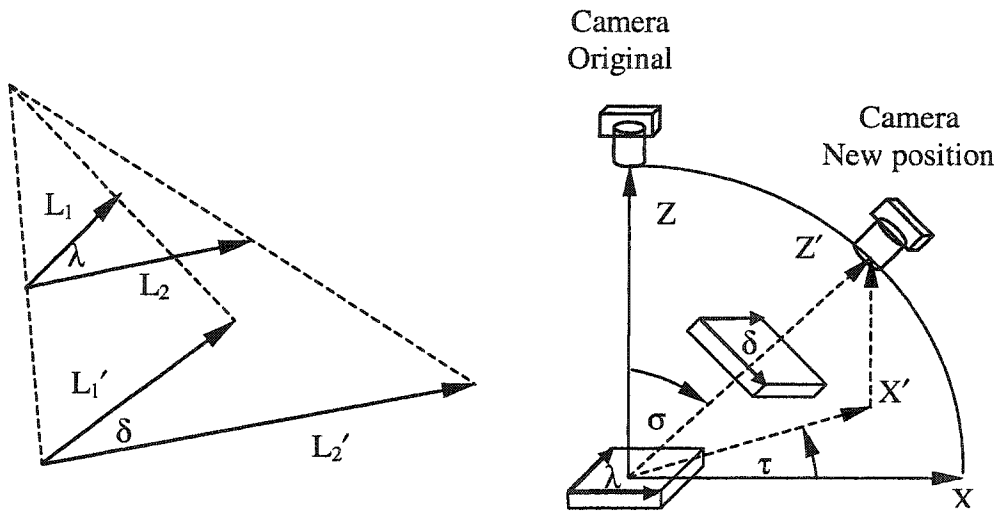
This section discusses probabilistic methods for estimating the pose of a 3D object.

Orthographic Projection: We use orthographic projection as an approximation for the actual central projection of the object onto the image plane. The implementation of the orthographic projection is justified in most practical cases where the distance between camera and object large in comparison to the object's dimensions.

3.3.1 Similarity Study: Effect of Viewing Angle Variations

While there is no affine or projective invariant for general three-dimensional point sets (Clemens & Jacobs, 1991), the deviations of angle and length ratios remain small for small angle variations in the neighborhood of the reference model. Therefore, it is possible to assume invariance with an acceptable error margin within small variations of viewing angle, using loop similarity. Values such as included angles between vectors or the ratio of lengths of two vectors vary only by a small amount within a limited range of viewing angles. This knowledge allows the discarding of matches between image and model groups with a low likelihood of being in actual correspondence. The probabilistic similarity method assumes that angles and ratios of distances between points in the model do not vary much when projected onto the image as the viewpoint varies over a limited range of the 3D viewing sphere (see Figure 3.6). In Figure 3.6 (a), there are two loops. One is reference loop from model (L_1, L_2, λ) and the other one is test loop from test scene

(L_1', L_2', δ). And we evaluate ratio of inner angle ratio and edge length ratio, respectively. Then we examine their change of deviation when the viewing angle varies. Figure 3.6 (b) shows such occasion. Here we have two viewing angle changes, azimuth and elevation, respectively. The key idea is we will try to find the range of viewing angle variation until the ratio of inner angle and edge length remain less changed.



(a) Loops from model and test (b) Viewing angle change in 3D

Figure 3.6 Similarity analysis on loop for 3D object

This type of evaluation has done to each model image when we build model database and based on that evaluation results we define probabilistic density function in terms of exponential form and plot the probability density curve and set up the acceptable scope of viewing angle change. This curve-fitted plot gives us basic idea that what interval of viewing angle is acceptable for given model object. In most case of polyhedral

objects, we find 15° to 20° degrees is acceptable. The probability density function $f(x)$ of angle and length ratio for the change of viewing angle is defined as follows:

$$f(x) = a_1 \exp(-a_2|x|) + a_3 \exp(-a_4|x|) \qquad (3.4)$$

where x means either length ratio $\left(\frac{L_1 L_2'}{L_1' L_2}\right)$ or angle ratio $\left(\frac{\delta}{\lambda}\right)$, and a_1, a_2, a_3, and a_4 are coefficients.

The resulting probability density functions have a strong peak similar to a Dirac delta function (Unit Impulse Function) at the preprojection (model) value. The values of the probability density function remain large in the neighborhood of the preprojection (model) value, so that we can build a probabilistic indexing system from a finite number of point sets in 3D.

Following the procedures of Ben-Ari (1990), Burns *et al* (1993), and Olson (1995), we first build model object groups that consist of a series of individual images taken at small viewing angle intervals. We compute probability density functions, comparing angle and length ratios of those model object groups with those contained in the test image. As we saw in Figure 3.6 above, it shows two indexing vectors forming a loop. We compare the magnitude of the two vectors and their included angle (Model in data base: L_1, L_2, λ), (Test scene: L_1', L_2', δ) respectively. First, we compare both loops with respect to change of viewing angle. As seen in Figure 3.6 (b) the elevation angle, denoted by σ, is the angle between frames Z and Z'. The azimuth angle τ is the rotation angle of the frame X to X'. In position 1 ($\sigma = 0$), the 3D object appears as a two-dimensional quadrilateral shape independent of τ, which means 2D invariant, whereas in position 2, the 3D polyhedral object's appearance varies with respect to every azimuth

angle change of τ. Using the trigonometric analysis of the relationship between angles δ, λ, σ, and τ in Figure 3.6 results in equation (3.5). The equation (3.5) tells us the how the two loops look alike when the viewing angle varies and is plotted for $\delta = 45°$ in Figure 3.7.

$$\lambda(\delta,\sigma,\tau) = \arctan\left[\frac{\cos\sigma\tan\delta}{1-\frac{1}{2}\sin^2\sigma\left(1-\frac{\cos 2\tau}{\cos\delta}\right)}\right] \qquad (3.5)$$

where λ inner angle of model loop

δ inner angle of test loop

σ elevation angle of camera

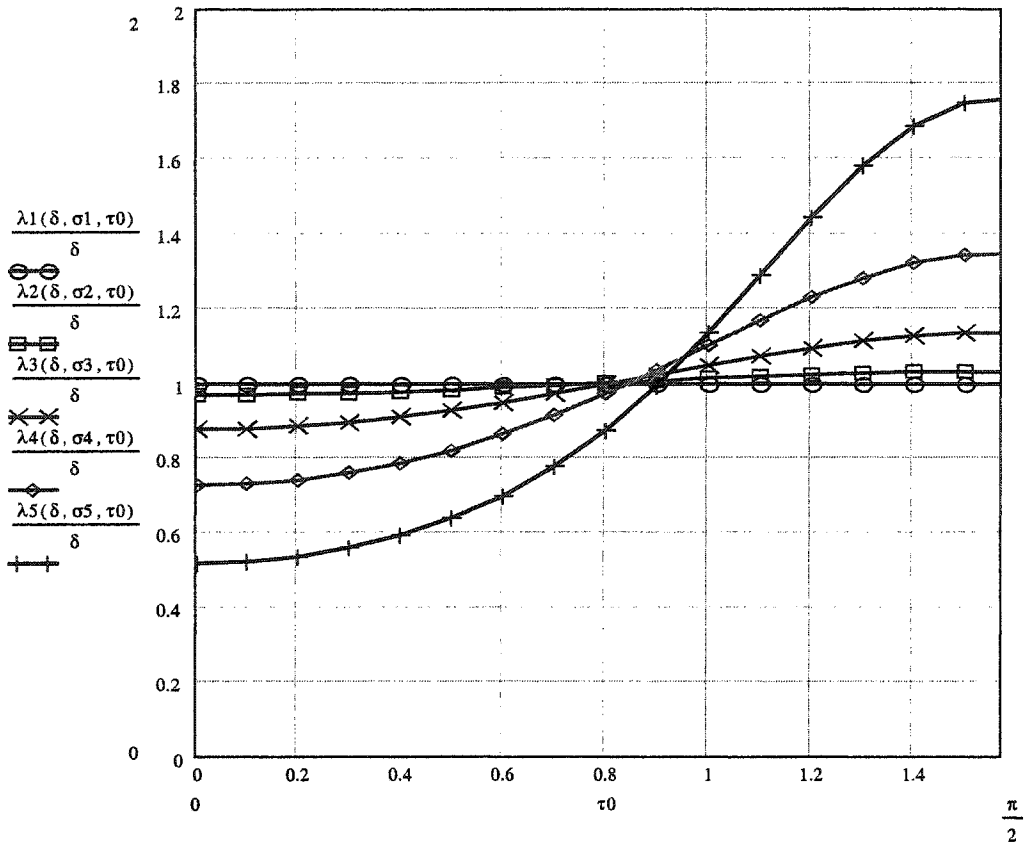
τ azimuth angle of camera

The scope of the abscissa for Figure 3.7 ranges from 0 to 75° at equal interval spacing of 15-degrees. From Figure 3.7, we note that the deviation of λ from its origin δ as a function of τ increases with σ. For small values of elevation angle σ and azimuth angle τ, below 30°, the indexing vector scale is almost the same as the 2D reference at $\sigma = 0$.

3.3.2. Recovery of Angle Ratio

The *angle ratio* is defined as the ratio λ/δ, where λ is the include angle of the model loop, and δ is the included angle of the test loop. It is convenient to define probability density function to examine the variation of angle ratio λ/δ since our goal is to find acceptable region of viewing angle that we could consider as good as invariant. And, as we have

seen in Figure 3.7, the variation of angle ratio effect shows exponential trends. So we define the probabilistic density function as in terms of exponentials in Equation (3.6).



Trace1: ooo at σ = 0 Trace2: at σ = 15 Trace3: xxx at σ = 30° Trace4: ◊◊◊ at σ = 45° Trace5: +++ at σ = 60°

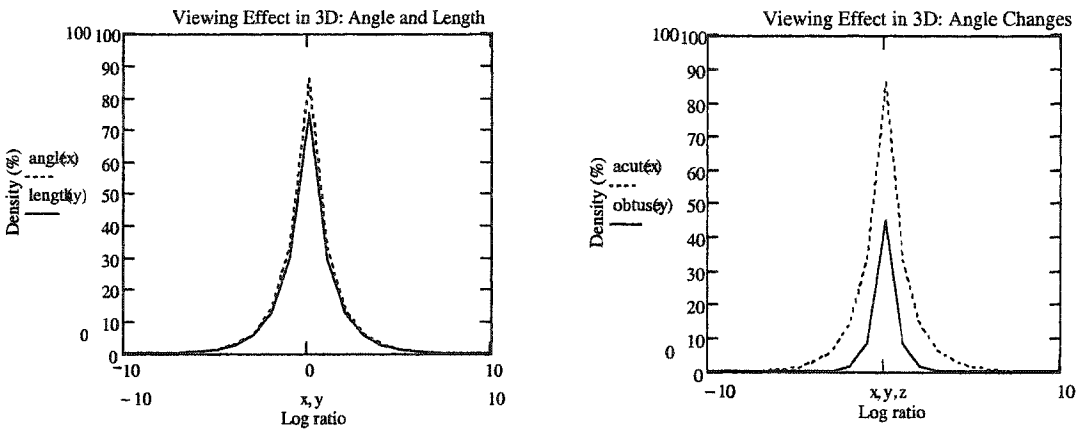
Figure 3.7 3D viewing effect with respect to elevation angle σ changes at $0^{\circ} < \tau < 90^{\circ}$ (Variation of angle ratio λ/δ at $\delta = 45°$)

$$f(x) = a_1 \exp(-a_2|x|) + a_3 \exp(-a_4|x|) \qquad (3.6)$$

where x is angle ratio or length ratio in logarithm scale a_1, a_2, a_3, and a_4 are coefficients.

The plot of Equation (3.6) shown in Figure 3.8 (a) for angle and length, and Figure 3.8 (b) for acute and obtuse angle in logarithm scale, respectively.

As we can see in Figure 3.8 (a), the deviations of angle ratio (dotted line) length ratio (solid line) and decrease with proximity to the x-axis in both directions from zero point, which is the 2D reference line. In Figure 3.8 (b), the deviations of acute angle ratio and obtuse angle ratio have been compared. And the change of acute angle of an image shows more sensitive variation than that of obtuse one. This explains the similarity information of angle and length shows almost same accuracy for the same loop, however, for the acute loop shows better results than obtuse one.



(a) Loop with angle and length change (b) Loop with angle changes

Figure 3.8 Variation curve showing angle and length ratio

The curves depicted in Figure 3.8 describe the matching accuracies. The sharp peaks of the densities at log (*ratio*) = 0 verify that matches are perfect at zero deviations from the model. For example, the probability for angle ratio change to fall within the range of $\delta/2 < \lambda < 2\delta$ is above 80% as we verified in Figure 3.7. Since the reason to change of loops angle between model and the test is only happens when the viewing

angle varies. Therefore this result supports the claim that *within a small range of variations of angles σ or τ, the deviations of length and angle ratios are small compared to neighboring loops in the database.* The more detailed probability density function of angle ratios can be approximated as below (Equation 3.7), which creates a peak curve (Dirac delta curve):

$$p_a(\log\frac{\lambda}{\delta}) = x_1\exp(-\frac{x_2}{x_1}|\log\frac{\lambda}{\delta}|) + x_3\exp(-\frac{x_4}{x_3}|\log\frac{\lambda}{\delta}|) \quad (3.7)$$

where p_a denotes the approximate probability density. The coefficients x_1 to x_4 (for the wide range of $-10 < \log(\lambda/\delta) < 10$) in Figure 3.8 (a) are as follows:

$$x_1 = 67.121: x_2 = 53.455: x_3 = 18.823: x_4 = 31.951: \quad (3.8)$$

Those coefficients x_i can be determined in statistical way after we examine the every angle ratio of model loop along with their viewing angle changes, and therefore this process would be exhaustive.

3.3.3. Recovery of Length Ratio

The estimate of the probability density of the projected length follows a technique similar to that employed in the previous section, by computing the probability density of a loop's projected length ratio. Orthographic projection is used again to approximate the actual central projection. Since the scale is unchanged for all viewing angles (at constant distance), it is not taken into account. We denote the model loop as the reference vector set $\{L_1, L_2\}$ and the test loop by $\{L_1', L_2'\}$. The ratio t is defined as $t = \frac{L_2'L_1}{L_2L_1'}$. Since the orthographic projection is linear, the lengths of L_1 and L_2 do not influence the ratio t. We approximate the density of t by the function $p_b(\log t)$ as

$$p_b(\log t) = x_1 \exp(-\frac{x_2}{x_1}|\log t|) + x_3 \exp(-\frac{x_4}{x_3}|\log t|) \qquad (3.9)$$

Equation (3.9) is already plotted in Figure 3.8 (b), which shows a sharp peak at $t = 1$ where t is the length ratio $\frac{L_2'L_1}{L_2L_1'}$. The coefficients $x_1$to x_4 are computed by a procedure that uses a minimum square error as the optimizing criterion. The optimal coefficients are based on results from Figure 3.8 (b) given as follows:

$$x_1 = 80.069: x_2 = 56.749: x_3 = 14.958: x_4 = 29.386: \qquad (3.10)$$

3.3.4 Complexity of the 3D Model Database

The probability distribution of individual features can guide the matching process that underlies recognition. Features whose presence is most strongly correlated with those of the test object can be given priority during matching. Features with the best *localization,* i.e. the closest match with model loops in data base, can contribute the most to an estimate of the object's position, while features exhibiting larger deviations from a model can be searched over large domains within the database. The advantage of creating a model image database using only a finite number of views lies in the reduction of data base size. This data base size reduction must be weighed against the corresponding decrease of probability of correct matching. Spacing the stored views at 1-degree intervals, for instance, would result in a total of 64800 (360 azimuth × 180 elevation for hemispheric view) model images. However, to build a hash table for such a model database (merely describing a single object) would be exhaustive manual labor. The preceding similarity analysis shows minor errors within 15° intervals, thus the hash table for the model database can be based on only 288 images.

3.4 Summary of Feature-Based Geometry Modeling Concept

The feature-based concept represents a 2D or 3D geometry by means of topology vectors and application of invariant indexing. The topological entities shown in Figure 3.9 are our choices of feature attributes. The topological entities allow a compact representation of more comprehensive structures in terms of connectivity structures. During model data base creation, junctions in the image are matched exactly with junction types in the network, that is, the model assumes reliable classification of image junctions.

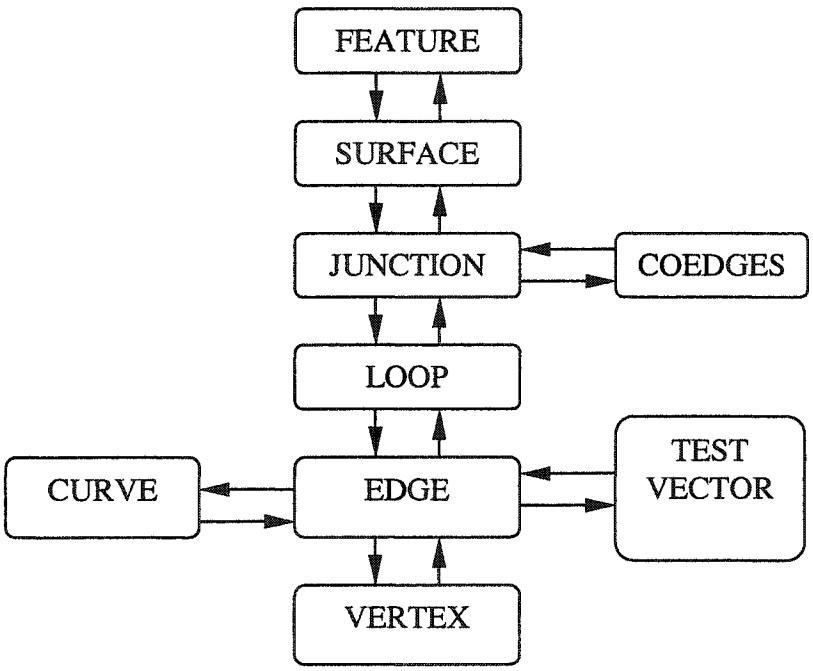


Figure 3.9 Topology of feature-based indexing

The use of feature-based geometry results in those advantages below: First, it is much easier and simpler to build model an image database and compare with test vector

groups since every attribute of feature shape is classified as topology, therefore if match has been detected then we can quickly find their candidate data.

The second merit is it is simpler to build modules for rule-based system because their attributes are classified as block data format. This advantage becomes important for fully integrated automatic target recognition. Recognizing object from the test scene by rule-based method requires compact modules as possible. And this compactness can be achieved from this feature-based concept.

CHAPTER 4

A KNOWLEDGE BASE FOR OBJECT RECOGNITION

This chapter describes the methodology for matching a model in the database with a shape in the test scene, using the feature-based concept. A knowledge base verifies the recognition hypothesis and re-orders the topology.

4.1 Introducing the Knowledge-Based System

A rule-based object recognition system contains a knowledge base and a set of algorithms or rules that infer new facts from knowledge and from incoming data. The *inference engine* sorts the data according to the rules, and attempts to reach new conclusions. Figure 4.1 shows a schematic diagram of the rule–based recognition concept. Data and rules are entered at the beginning, and iteratively updated during the recognition process. The most important elements of the rule-based system are the decision functions that make judgment in the inference engine.

Knowledge-based systems (KBS) are computer programs designed to simulate the problem-solving behavior of human experts within narrow domains or scientific disciplines. The theory of knowledge bases is a sub-set of Artificial Intelligence (AI). In most KBS development environments, the rules as well as data can be entered in any order without affecting the system's conclusions.

Often new sets of rules can be added without requiring a reconfiguration of the knowledge base. In the context of this analysis, two methods of inferencing are applied: *forward chaining* and *backward chaining*.

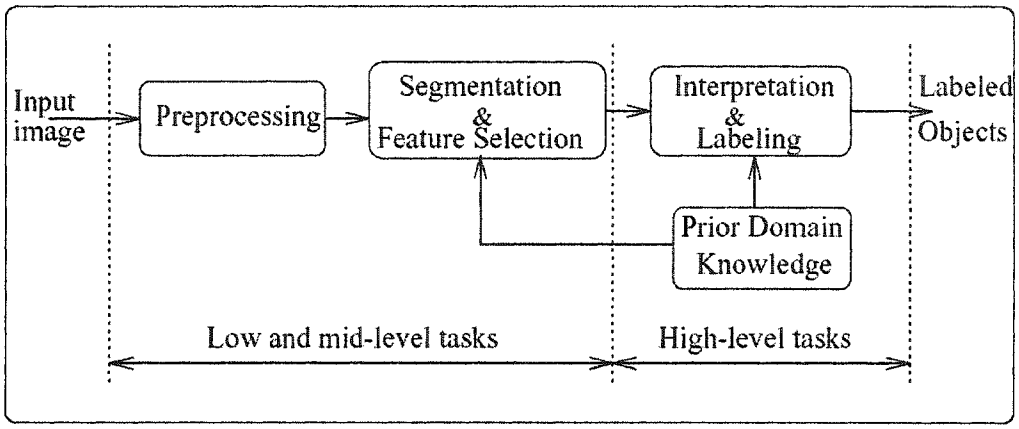


Figure 4.1 Knowledge-Based Processing in an Object Recognition System (quoted from Aggarwal, 1996)

Forward chaining is a data-driven technique used in constructing goals or reaching inferences derived from a set of facts. This method is also known as *bottom-up reasoning*, *data driven reasoning* and *antecedent reasoning*. It searches systematically through the rule base for rules whose conditions are true. This mode of reasoning is the dominant approach for hypothesis verification in this paper. Forward chaining is implemented as a set of if-then rules.

The *Backward-chaining* algorithm proves a goal by recursively breaking the goal down into sub-goals and trying to prove these until facts are reached. Facts are goals with no sub-goals, which are therefore always true. Backward chaining is the programming

mechanism used by logic programming languages such as Prolog. Backward chaining is also known as *top-down reasoning*, *good-based reasoning*, and *consequent reasoning*. There are conditions when the order of the rules and facts may influence the results. Here, we implement backward reasoning in the form of whenever-do rules.

KB systems for pattern recognition are generally hierarchically structured. At the lowest level of hierarchy, the data structures represent edges, junctions, and loop information. The application of KB system rules also identifies possible regions of matches and possible groupings of regions, which are themselves grouped into models representing views of objects. Most rules are general and apply to all data. Most rules employ forward chaining, and implementing a prediction/hypothesis/verification paradigm. The elimination of non-matching data sets occurs through a rule of constraint propagation. The KBS described below was developed in CLIPS (C Language Integrated Production System). A detailed discussion of CLIPS follows in Section 4.3.

4.2 Related Work

In recent years, many authors applied knowledge-based reasoning to the object recognition problem, using model based object recognition. Biederman (1985) proposed the theory that a particular vocabulary of components, called *geons*, derived from perceptual mechanisms and an arrangement of these components can be used to represent an object. Experiments on human perception of briefly presented pictures have shown that perceptual recognition of objects can be viewed as a process in which the image is segmented at regions of deep concavity into an arrangement of geons. A geon is derived from the contrasts of five properties of edges in a 2D image: curvature, collinearity,

symmetry, parallelism, and cotermination. Indeed, identifying the arrangement of a few geons from a limited set is enough to recognize tens of thousand of objects.

Edge-based representation of a few simple components has been shown to suffice for initial access to a mental representation of an object model. For real objects, two characteristics contribute to the object's classification: One is surface color and the other is contour (edge). Edge-based representation is also called geometry-based representation and is the basic idea of the KB system presented here. Shapiro *et al* (1984) proved the results of psychological experiments performed by Biedermann. They implemented a relational model for describing 3D objects. This model was used for initial matching with an unknown object. It uses sticks (long, thin parts), plates (flat, wide parts), and other simple geometric elements to describe the parts of an object. The authors define a measure of relational similarity between 3D object models as well as a measure of feature similarity, both based on Euclidean distance norms between attribute-value tables. Their results suggest that relational similarity is much more powerful than feature similarity and should be used when grouping objects in the database for fast access.

Lowe (1987) implemented a system that can recognize 3D objects from unknown viewpoints in single gray-scale images. He points out that human vision does work very well at recognizing images, such as simple line drawings that lack any reliable clues for the reconstruction of depth prior to recognition.

Robert (1965) and Guzman (1969) developed an algorithm that groups polygons using local evidence which inserts links at junctions between two regions if they belong to the same body. Vertices are classified into types, and each type always belongs to the same type of polygon.

4.3 Expert System Implementation: CLIPS

4.3.1 History of CLIPS

The C Language Integrated Production System (or CLIPS) dates back to 1984 at NASA's Johnson Space Center. At this time, the Artificial Intelligence Section had developed over a dozen prototype expert systems applications using state-of-the-art hardware and software.

The original intent of CLIPS was to gain useful insight and knowledge about the construction of expert system tools and to lay the groundwork for the construction of a replacement tool for the commercial tools then in use.

4.3.2 Reason for using CLIPS

While many expert system shells and tools are commercially available (Giarrantano, 1998), CLIPS is available as C-source code, and is thus completely transparent to the programmer. CLIPS has been successfully applied in several real-world applications. Other benefits of using CLIPS include:

Rapid Execution

(1) All rules are active independent of the data base search status, since all rules are connected to the current active node.

(2) CLIPS can control the propagation process by explicitly invoking or disabling parts of the process, so that only a few propagations are needed to update the values in the whole network.

Convenient Use

(1) The data values associated with the rules can be of any well-defined data type, not just probabilities.

(2) CLIPS allows the programmer to define his/her own inference method, add new knowledge, and delete old knowledge, as well as to modify existing knowledge easily.

(3) CLIPS facilitates the automatic generation of program structures for object representation.

(4) CLIPS provides a powerful debugging tool that allows the user to pause and trace the value changes of one or more nodes in the network during value propagation. It also allows the user to examine the database.

(5) Since it is formulated in C, a programmer can understand the program CLIPS thoroughly.

4.4 Rules and Decision Functions for Probabilistic Reasoning

Here we introduce the rules and their interpretations used in the recognition process. The decision functions have been implemented in CLIPS.

Rule 1: Edge and Test Vector Match

Let *VIEW_M* be a view type of a model image *M*, and let *T_VECTOR* be the set of test vectors in any given test scene *T*, and let *EDGE_VIEW_M* be the set of magnitudes of edges from *VIEW_M*. Then an instance of *EDGE_VIEW_M*, assigned to *T_VECTOR*, *INST1*, becomes

$$INST1: T\_VECTOR \rightarrow EDGE\_VIEW\_M$$

The decision function of rule one is the comparison of the edge magnitude between model image *VIEW_M* and test vector from test scene *T* as follows:

Decision function for rule one:

$$if\left(\left|\frac{EDGE\_VIEW\_M}{T\_VECTOR}\right| \leq tol\right) then(T\_VECTOR \rightarrow T\_CO\_EDGE) \qquad (4.1)$$

Interpretation of rule one:

Rule one collects co-edges from test vectors during the recognition process by matching their magnitude with edges from the model image. As in the previous chapter, the test vectors are drawn by connecting the vertex in the test scene, and then compared with model edge data. The co-edges are those test vectors the have matches during this comparison. After the co-edges have been found, they are defined as junctions. A junction is a set of edges or test vectors having correct magnitude matches. Loop classifications are performed by rule two.

Rule 2: Loop Search

Let loops from a view type of model image *VIEW_M* be defined as *LOOP_VIEW_M* , and let *T_LOOP* be the set of loops in test scene *T*. Then an instance of *LOOP_VIEW_M*, assigned to *T_LOOP, INST2,* becomes

$$INST2 : T\_LOOP \rightarrow LOOP\_VIEW\_M$$

The decision function of rule two is the comparison of inner angles between *LOOP_VIEW_M* and *T_LOOP* after the collection of junctions from the test scene.

Decision function for rule two:

$$if\left(\frac{\angle LOOP\_VIEW\_M}{\angle T\_LOOP} \le tol\right) then(T\_LOOP \rightarrow LOOP\_VIEW\_M) \qquad (4.2)$$

Interpretation of rule two:

Two interpretations take place in rule two. One is to find loops from the co-edges. The second one is their reorganization, using their connectivity so as to evaluate the shape of the surface or super perimeter. From a geometric point of view, the loop search seeks to find congruity of triangles; after all, a loop represents a three-segment convex geometry. If a match based on prior knowledge from the model image is found, those matching loops are connected as a surface.

Rule 3: Verify Surface Invariance

Let all the transformation information about the surface from a view type of model image *VIEW_M* be defined as *SURFACE_VIEW_M*, and let *T_ SURFACE* be the set of all transformation information about the surface of test scene *T*. Then an instance of *SURFACE _VIEW_M* to *T_ SURFACE, INST3,* becomes

$$INST3: T\_SURFACE \rightarrow SURFACE\_VIEW\_M$$

The decision function of rule three is a comparison of surface transform invariance between model image and test scene (refer to Figure 3.3 and equations (3.1) and (3.2)).

Decision function for rule three:

$$if\left({}^{U}_{A}G^{A}Loop = {}^{U}_{B}G^{B}Loop\right) then(T\_SURFACE \equiv SURFACE\_VIEW\_M) \quad (4.3)$$

Interpretation of rule three:

Even if we have a perfect match of loops between model image and test scene, there is still a need to verify their match by comparing surface invariance. Some of the reasons to verify surface invariance are the followings:

(1) Symmetry: 3D objects with symmetries with respect to x, y, or any other principal axes will have the same loop matches.

(2) False positive: Due to overlapping or image noise, misunderstood loop matches can occur when unwanted interesting points are entered into the analysis.

(3) Over determined problem: Certain recognition cases can become over determined. Over determination occurs when the number of matched loops in the test scene exceeds the number of a model object's loops in the model database. The effect can result from the cumulative errors from data collection, as well as from probabilistic viewing analyses in 3D. Such errors create false positives and create "unwanted" loop matches.

To resolve these problems, we must store in the model image database not only magnitude and angle information, but also transformation invariance information about the surface. As discussed in chapter three, every loop in the model image database contains surface invariance information in the form of the invariance vector that connects the middle points of two loops. Rule three examines these vectors matches after having found loops from co-edges.

Rule 4: Verify Feature Neighborhood

Let any super perimeter from a view type of model image *VIEW_M* be defined as *SUPER_VIEW_M* and let *T_ SUPER* be the super perimeter in the test scene *T* as newly found. Then an instance of *SUPER_VIEW_M*, assigned to *T_ SUPER*, *INST*4, becomes

$$INST4 : T\_SUPER \rightarrow SUPER\_VIEW\_M$$

Decision function for rule four:

$$\begin{array}{l} if\left(T\_LOOP \supseteq SUPER\_PERIMETER\_VIEW\_M\right) \\ then\left(T\_SUPER \rightarrow FEATURE\_VIEW\_M\right) \end{array} \quad (4.4)$$

Interpretation of rule four:

To recognize 3D model features from the test scene, we use a super perimeter search algorithm. As discussed in the previous chapter, the super perimeter search enhances the reliability of the matching process, and reduces the number of required iterations. Since we are mainly handling 3D shapes projected onto plane surfaces, the super perimeter definition for object recognition is a convenient tool. During recognition, whenever a super perimeter is recovered, unrelated candidate data sets are quickly removed from the search domain.

Rule 5: Parallel Process and Backward Chaining

For 3D object recognition, strategies in addition to the super perimeter search are available. A surface may exhibit contour features useful for identification even when the super perimeter search has failed. Distinctive surface features not characterized by the super perimeter may appear prior to lower rule execution or at any point thereafter. Such additional instances of distinctive patterns reduce the number of required iterations in the

recognition process. We define a *whenever-do* rule for dealing with such occasions. Let any feature data belong to model image *VIEW_M* as *LOOP_VIEW_M* and let *T_ LOOP* be the set of surfaces in the test scene *T* as newly found. Then an instance of *FEATURE _VIEW_M*, assigned to *T_ FEATURE*, *INST*5, becomes

$$INST5 : T\_LOOP \rightarrow FEATURE\_VIEW\_M$$

Decision function for rule five:

$$\begin{aligned} &whenever(T\_LOOP \supseteq SUPER\_PERIMETER\_VIEW\_M) \\ &do(T\_SURFACE \rightarrow FEATURE\_VIEW\_M) \end{aligned} \quad (4.5\ a)$$

$$\begin{aligned} &whenever(T\_LOOP \supseteq SURFACE\_VIEW\_M) \\ &do(T\_SURFACE \rightarrow FEATURE\_VIEW\_M) \end{aligned} \quad (4.5\ b)$$

Interpretation of rule five:

The loops belonging to the super perimeter and surface appear in no particular order. Therefore, whenever loops belong to a surface or super perimeter, we choose the related candidate images as the best data sets without considering the sequence of iteration. Figure 4.2 explains parallel searching for super perimeter and surfaces. The solid line shows the normal search algorithm by if-then rule, whereas the double-dot dashed line shows the whenever-do rule search. Backward chaining becomes an economic tool for 3D recognition, since 3D objects are analyzed in terms of their 2D projection. Therefore to recognize and recover 3D object from its 2D image implies the simultaneous detection of loops from the super perimeter as well as those within surfaces. Whenever a loop from the test scene matches a loop of the super perimeter in the model, we start the feature search, along with a surface search using backward chaining. Both forward and backward

search algorithms are executed in parallel; the iterations end whenever a perfect match is found.

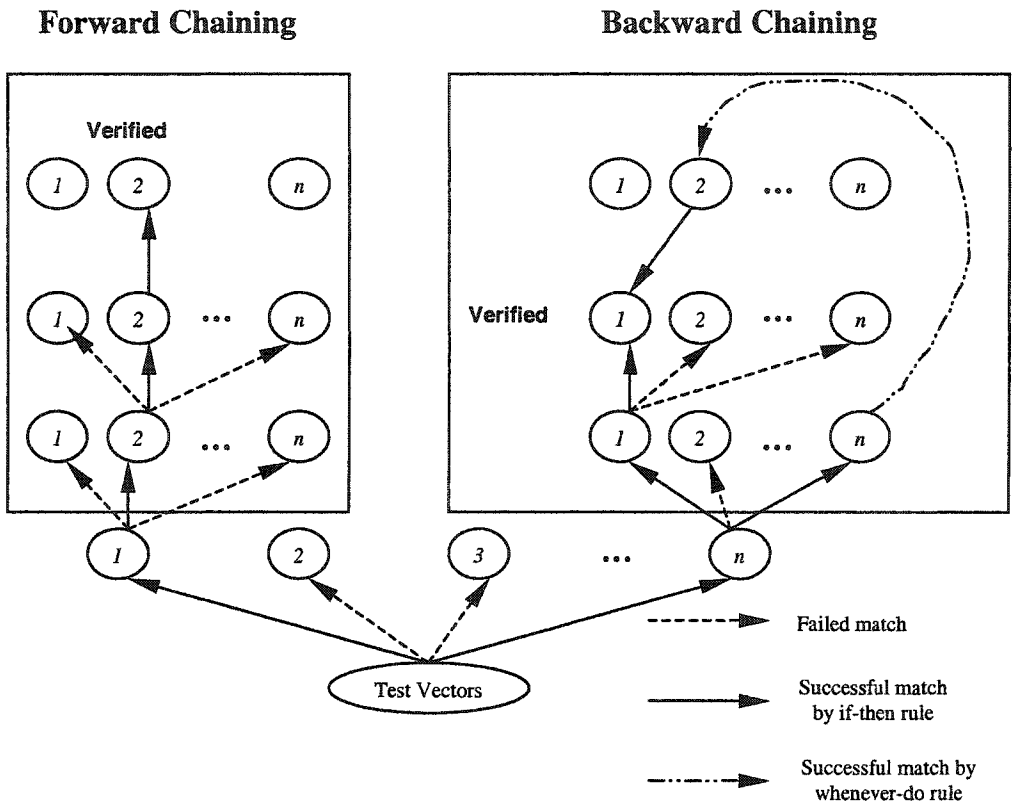


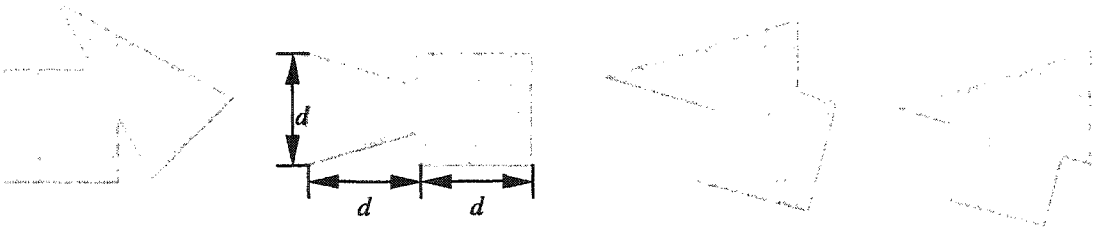
Figure 4.2 Network representation of parallel processing in recognition system

4.5 Occlusion

4.5.1 Definition

In the context of feature-based indexing, using vector geometry and loops for object matching, occlusion is defined in terms of loop relations: Occlusion occurs if *one or more loop(s) from a test scene is damaged* or *missing when compared to the object's model image loop(s)* (see Figure 4.3). For example, Figure 4.3 shows four images of two objects

overlapping. One is a triangle with three loops, while the other is a square with four loops. The triangle in Figure 4.3 (a) still has its three vertices; therefore it is not occluded since we can generate all associated loops. The square, however, is missing one of its loops. Two other loops are damaged by the triangle; therefore the square is partly occluded. The square object can be recovered by probabilistic reasoning.



(a) square is overlapped | (b) Triangle is overlapping | (c) square partly occluded. | (d) only triangle can be recovered.

Figure 4.3 Four Cases of overlapping

Figure 4.3 (b) shows a special case of occlusion. The triangle is overlapped by the square, but the dimensions of the remaining shape are similar to that of the square. Unless additional data (loops) characterizing the triangle exist, it is not only impossible to recover the triangle, but also since the occluded triangle resembles the square, the number of loops increases beyond four, leading to an over determined problem as we discussed earlier.

The loss of the triangle's third vertex creates damage to the associated loops. Occlusion cannot be decided based on missing vertices or damaged loops. There must be a sufficient number of intact loops remaining for occluded shape recovery. Figure 4.3 (c) shows another example for loss of a vertex and its associated loops from the triangle.

The triangle has only three vertices; therefore we can only recover the line between the two visible vertices. A single line cannot be reliably associated with an object. Therefore, in the absence of additional data, the triangle cannot be identified. For the square a third undamaged loop exists, therefore the square is partly overlapping and can be recovered. If there is no additional loops are found in the test scene, the algorithm decides that no matching model image in the test scene exists. In Figure 4.3 (d), the square lost all additional loops and the triangle still contains all loops. Therefore the algorithm identifies the triangle, but not the square. Based on the preceding discussion, an alternate definition of occlusion is:

Definition of Occlusion (II): *If during the comparison of any model image with the test scene any loop of a candidate object is missing or damaged, then it is considered as possibly occluded.*

The definition above presents a necessary condition for overlapping. The level of confidence for concluding overlap depends on the number of remaining intact loops. The more intact loops exist, the more accurate is the determination of overlap and the more reliable is the object identification.

4.5.2 Defining the Degree of Uncertainty (DOU)

Since overlapping is defined as loss or damage of loops, the degree of uncertainty is defined based on the number of intact loops during the comparison between model image and test scene as follows:

Rule 6: Degree of Uncertainty (DOU)

$$DOU = t - m$$

where t : number of intact loops in the test scene

m : number of loops in the model image

Then an instance of *DOU, INST6*, becomes

$$INST6 : DOU \rightarrow OCCLUSION$$

Decision function for rule six:

$$\begin{array}{l} if(DOU \subseteq OCCLUSION) \\ then(T\_FEATURE \rightarrow OCCLUSION) \end{array} \quad (4.6)$$

Interpretation of rule six:

The evaluation of DOU is key to verifying the presence of occlusion in the test scene. The difference between the number of detectable and missing (or defective) loops indicates whether recovery is possible and the degree of reliability of the resulting identification. If the value of DOU becomes reaches -m (the number of model image loops, which means no match), then no identification is possible. Therefore the possible range of recognition for overlapping objects lies within $-m < \text{DOU} < 0$.

The process is illustrated through the examples in Figure 4.3. After completion of the loop search recognition requires the DOU value to determine overlapping and recover missing or damaged loop(s), until the DOU value becomes zero. Table 4.1 lists the possible values of DOU. If the DOU value becomes negative, the match problem is underdetermined, and the test image contains fewer loops than the number of loops

describing the model. Under determination can also arise for reasons other than occlusion.

Table 4.1. Variation of DOU

DOU > 0	Over determined
DOU = 0	Exact solution
-m < DOU < 0	Underdetermined (overlapping); possible to recover
DOU <= -m	Underdetermined (overlapped); impossible to recover

A negative value of DOU merely implies loss or damage of loops in the test scene when compared to the model data. Under determination is common in 3D real images, due to image noise in the test scene. Therefore the DOU value is an important tool for deciding whether all loops have been recovered, to check for false positives, and to determine the end of iterations. Table 4.2 has DOU values in the examples in Figure 4.3.

Table 4.2. DOU variation and its interpretation in Figure 4.3

Figure 4.3	DOU	Remarks
(a)	0 for Triangle, −3 for Square	
(b)	−3 for T, 4 for S	Two squares are found (over determined) and regenerated and impossible to recognize triangle.
(c)	−3 for T, −1 for S	Impossible to recognize triangle
(d)	0 for T, −4 for S	Impossible to recognize square

4.5.3 Probabilistic Reasoning to Recover from Occlusions

To recover a damaged loop, we employ probabilistic reasoning. Prior knowledge from the model image database includes surface and feature information belonging to each loop. We connect the related loops that belong to the model feature database until a perfect feature match with the test scene results. Table 4.3 shows the rules used in the process.

Table 4.3 Rules and interpretation for pattern recognition

Rules	***Attribute***	***Function***	***Order of Sequence***
Rule 1	Rule of edge	Classifying junctions from co-edges	Bottom level
Rule 2	Rule of loop matching	Search for the loops	Intermediate level
Rule 3	Rule of surface invariance	Confirm the surface invariance	Rules 3 through 6 are evaluated iteratively in parallel
Rule 4	Rule of feature neighborhood	Super perimeter search	"
Rule 5	Rule of parallel processing and backward chaining	Parallel process and backward chaining	"
Rule 6	Rule of defined occlusion	Determine occlusion by degree of uncertainty (DOU)	"

4.6 Assessment of Probabilistic Reasoning

To evaluate the entire recognition process with regard to its ability to identify a shape from any viewing angle, we first consider a bottom-up, or data-driven, recognition scheme. This scheme first evaluates the edge and junction type information in the image to establish a complete set of loops. Until now we have assumed exact matching with the

unknown edges and junctions in the test scene. As the bottom-up recognition scheme moves through *rule two*, it generates at each level set of alternative hypotheses, and eliminates as many alternatives as possible given the information available at that level. This scheme generally results in much useless work because it does not look ahead to higher levels of rules. Otherwise, it could eliminate hypotheses that are unpromising from a more comprehensive point of view and it could defer decisions – and the work required to make them – until more critical decisions have been made.

In contrast to a bottom-up, data-driven control scheme, a top-down, model-driven scheme does a small amount of work at low level, and then creates hypotheses. This hypothesis is more tentative than those made in a bottom-up scheme and is initially less constrained, hence more numerous. A top-down scheme thus has the disadvantage that decisions made at lower level rules are initiated by disputes at higher levels; in particular, work at rule level one is done only as required.

We have implemented a control scheme that combines bottom-up and top-down elements as occasioned by the random sequence of pattern searching. While both the bottom-up and top-down schemes inspect the loop elements at the intermediate level, the scheme proposed here proceeds according to its current state of knowledge, performing rule base matching when no connectivity is known, and arranging identified surfaces according to known geometrical relations contained in the model database.

Bottom-Level Analysis: The scheme extracts from the image the edge and junction information required to make reasonable low-level hypotheses. Taking only the best low-level hypotheses, it arrives at a restricted set of hypotheses at the intermediate level. These intermediate level hypotheses allow more focused decisions at lower levels;

these decisions in turn lead back to improved intermediate level hypotheses. The elimination of non-valid sets of intermediate level hypotheses also restricts the search domain at advanced levels. When a 3D feature has been identified, the advanced level hypotheses descend to intermediate and low levels; the resulting work at these lower levels produces further instances and resolution of conflicting hypotheses.

Intermediate Level Analysis: The structures of hypotheses by control scheme results from low-level search step up to both intermediate and advanced levels. At the intermediate level, the recognition process first focuses on proper match of loop types to model loops. This search will continue till any number of matches of loops found for all the remaining hypotheses from the low-level analysis. The focus thus expands to the neighbors of the model loops, which are classified in terms of the spatial relationships between the model loops and its junctions. If a loop matches with a neighbor of several loops from one model, then the recognition process focuses all those matches as well since classifying those matched loop would generally eliminate number of intermediate level hypotheses.

Advanced Level Analysis: At advanced level, a surface type is defined if enough of its connectivity types matches with correct matched loops from intermediate level search. The goodness of an instance's claim depends on the proportion of the surface's connectivity types that are assigned to model image and on the goodness of their matches. The recognition process focuses on good instances of surface invariance match; for each such instance, it tries to find matches for the connectivity type of loops in the corresponding surface and feature that have not yet been verified their connectivity in the image and correct matches will be filled in the "bucket" of positive recognition table. The

growth of instances and hypotheses in the advanced level search may follow many different patterns depending on how comprehensive the corresponding view types and the order in which constituent connectivity types are assigned. As instances grow, they contend for newly claimed surfaces in the image; such contention also directs the search strategy of the recognition process for surfaces and features – backward and forward chaining, for example.

At each step of verification, the level of confidence in the correctness of a match can be expressed in terms of the loop types and spatial relationships of the object's surfaces and features. The reliability of such matches depends on the viewing angle. To reduce the level of indeterminacy in an interpretation, the reliabilities of current matches with candidate model surfaces in the image must be ranked. Here, those instances will be classified by ranking the matches, i.e. by counting the number of additional validations for each hypothesis. At the final stage of evaluation, poorly ranked hypotheses will be rejected. The process terminates when no further disputes can be resolved. The recognition outcome is thus a set of interpretations whose cardinality depends on the number of resolved disputes.

There are considerable opportunities for parallelism in this hierarchical scheme. Any number of disputes can be evaluated in parallel at the intermediate level. In the parallel case, however, alternative hypotheses cannot be eliminated until the results of the individual parallel processes are complete; similarly, any number of instances can be evaluated in parallel, but contention for surfaces and features requires either communication between processes or waiting for results.

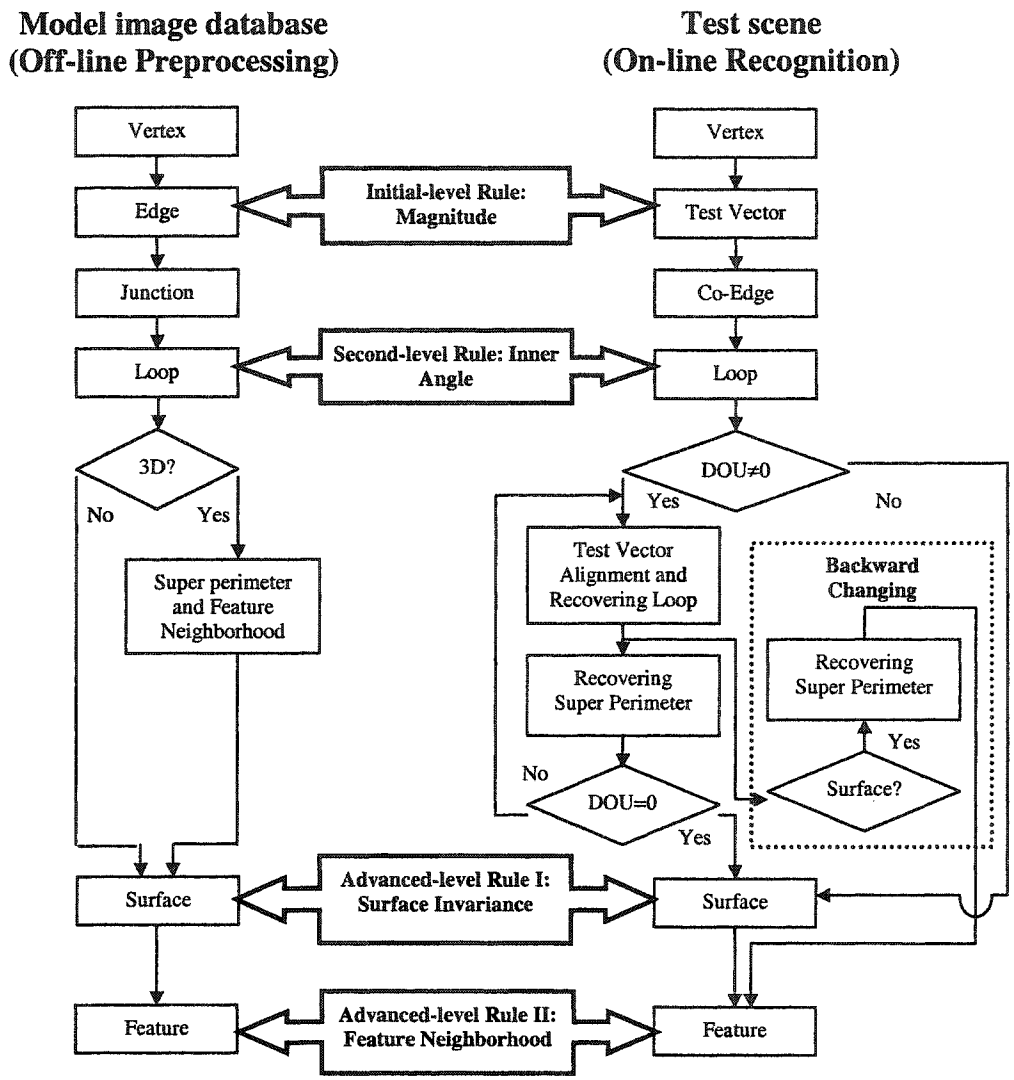


Figure 4.4 Knowledge based system for three dimensional pattern recognition

Finally, even intermediate and advanced level rule instances can be evaluated in parallel, but the propagation of the effects of one to the other requires a separate, sequential step. More complete parallelism is possible when the images are partitioned into several connected components in 3D. Figure 4.4 presents a flowchart of the entire recognition process. Chapter 5 presents the implementation of this chapter's algorithm, applied to gray-level images, and discusses the experimental results.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, the details of a complete recognition system, in which the indexing method had been covered, will be presented. The system contains all of the elements crucial to any model-based recognition system, including the model database itself (and choice of model representation), feature detection, and feature grouping, indexing, and matching (hypothesis verification). The matching choices for each of the sample images presented in this chapter are not unique, and the ability of the indexing system to generate valid hypotheses depends on image quality. The real CCD images presented below contain noise and varying levels of contrast. The analysis is able to address the majority of the issues that appear in realistic scenarios and perform correct identifications in most cases.

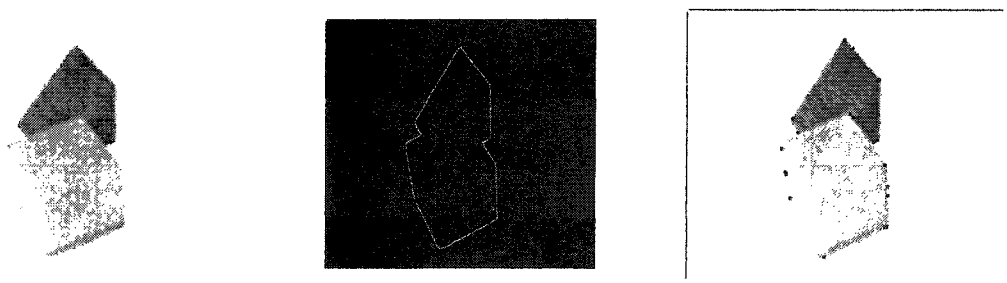
5.1 Grouping Data from Model Image and Test Scene

Models are polyhedral and are stored as formatted lists of points, edges, junctions, loops, and surfaces. Surface markings are also included in the representation; these are labeled to indicate the features. Test scenes, on the other hand, are processed only with regard to extracting the interesting points.

Model data are currently generated manually using a viewpoint-centered coordinate system. This is a time consuming process, therefore the model database available for experiments contains just seven 3D object shapes, each shape represented through 24 images taken at varying viewing angles. Most other polyhedral object databases reported in the literature consist merely of abstract geometric figures rather than data sets extracted from real images. The algorithm has been implemented in MathCAD and CLIPS.

5.1.1 Corner Detection

Corners are the fundamental components for building feature groups in terms of interesting points. These points are the main data type used to test the indexing algorithm performance. To detect corners, we first perform edge detection to find object contours in the image. From the several available contour detection algorithms, we selected the one by Canny (1986). Figure 5.1(b) shows the contours of the CCD image in Fig. 5.1(a).



(a) Original image (b) Contour image (c) Detected corners

Figure 5.1 Edge and corner detection from 3D real image

The corner detection algorithm identifies convex-to-concave or sharp junctions. Figure 5.1(c) shows the corners extracted from Figure 5.1(b). After obtaining the corners by applying a customary 5 × 5-pixel mask, we employ Cubic *B*-Spline smoothing to eliminate false corners created by image noise (Medioni & Yasumoto, 1987). The edge and corner detection algorithms are unable to detect contours characterized by only small grey-level variations, such as the boundary between the cube and the darker prism behind it. Even after smoothing, false positive corners are frequently still present (see Figure 5.1(c)).

Table 5.1 Indexing methodologies

Preprocessing stage (off-line):

Step 1: Extract interesting points from model image

Step 2: Calculate edges and loops data

Step 3: Evaluate surface invariance using transformation invariance of mid-point of loops

Step 4: Evaluate loop connectivity of super perimeter

Recognition stage (on-line):

Step 1: Extract interesting points and build test vector sets

Step 2: Pick one model image dataset and compare with test scene data to find co-edges and loops

Step 3: Evaluate DOU values and verify hypothesis

Step 4: Regenerate matching results

Step 5: Go to the step two, select next model image dataset and repeat the remaining steps till the last model

5.1.2 Search Path and Ranking

The evaluation of the DOU values proceeds according to a ranking system (Shimshoni & Ponce, 2000). The ranking system is similar to weighted k^{th} nearest neighbor or WkNN (Duda & Hart, 1973). Upon completion of the pattern matching process, a weighted value (or matching result with extra credit) is assigned when the loop belonging to a surface also belongs to the super perimeter. All matching results are stored with their rankings and sorted according to their ranking scores. This type of decision is termed 'best bin first' (or BBF) algorithm by Beis & Lowe (1998).

5.1.3 Recognition System Overview

Table 5.1 presents an overview of the entire recognition system. We distinguish between two main components, an offline preprocessing stage, and a real-time recognition stage.

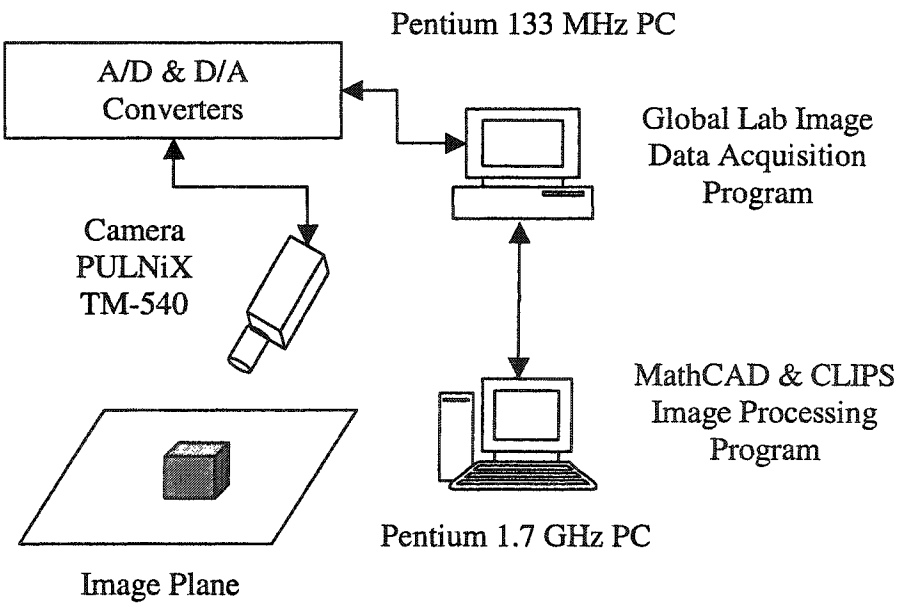


Figure 5.2 Computers and sensors used in the experiment

In our application a camera is used to capture the image of the object of interest, which yields the gray level image information after processing. The sensors are connected to the personal computer. The configuration is explained schematically in Figure 5.2.

5.2 Recognition Experiments on 2D Synthetic Images

5.2.1 Synthetic Data for 2D

2D synthetic images were created before 3D real image recognition for the purpose of algorithm development and initial testing. Synthetic images have the advantage of (i) easier algorithm inspection and verification (ii) significantly reduced noise effects, and (iii) easier and more accurate transformations to set scale, translation, and rotation information of the image. Figure 5.3 shows the set of 2D and 3D images used in our experiments. All experiments in this chapter follow the same basic procedure. First, a set of training images was generated, using either a tessellation of the view sphere, or random views sampled uniformly over the view sphere.

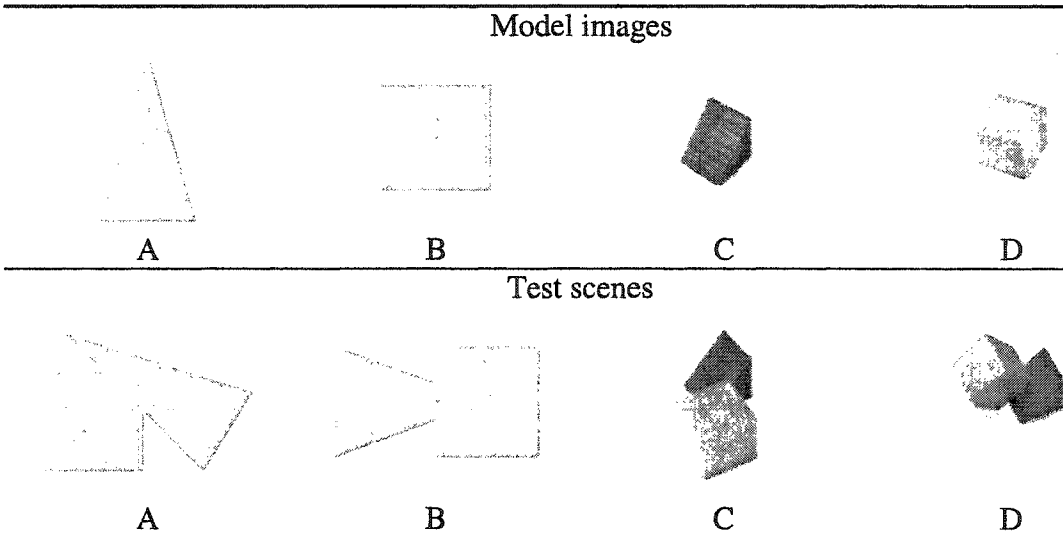


Figure 5.3 Model images and test scenes for recognition

Initially, real 3D test scenes contained a single database object, chosen at random and viewed from a random viewpoint. Gaussian noise filtering was applied to all images before edge and corner detection.

Two performance measures were applied as follows:

Success Rate: The relative frequency of finding at least one correct match after a complete analysis.

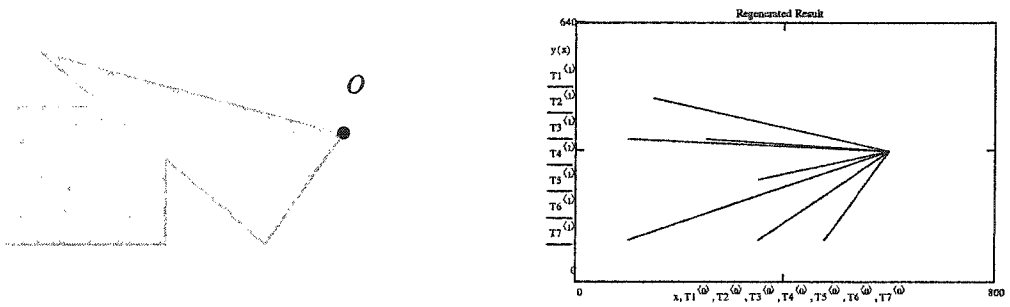
Index ranking: For correctly matched images, the type and number of correct hypotheses inferred during the analysis indicate the degree of confidence in the correctness of the identification, which is termed *index ranking*.

5.2.2 Experiment 1: Match Test Scene Object with Data Base Model

In this experiment, we verify the algorithm following the sequence of the rules of Chapter 4 for feature recognition. The procedure of reasoning is forward chaining.

(1) Generating the Test Vector

The first step in the recognition process is to draw the test vectors by connecting all interesting points to all others. Figure 5.4 shows one of test vector sets from point *O*. The dimensionality of the test vector for *n* points is *O* (n^2). We seek to remove duplicate test vectors, e.g. having computed vector *A* to *B*, vector *B* to *A* is not required.

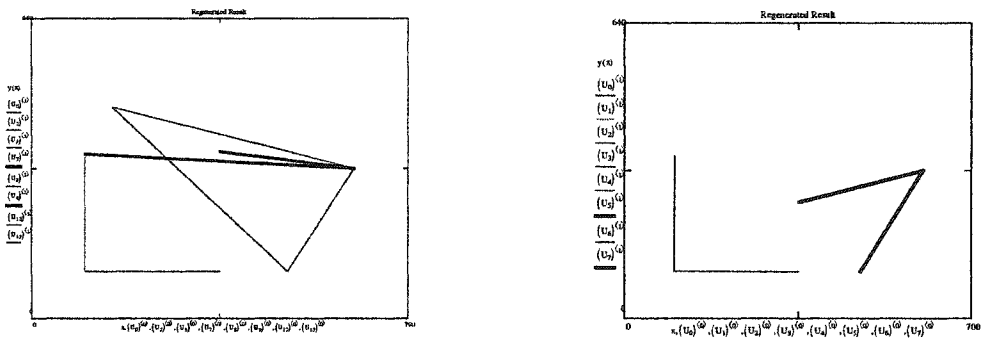


(a) Test scene A (b) Test vectors from *O*

Figure 5.4 One of eight test vector sets from test scene A at point *O*

(2) Find Edge Match and Collect Co-Edge Data

From the test vectors we match edge magnitudes between the test vectors and model edges by evaluating the first-level rule in the recognition process. Figure 5.5 (a) shows matching results for the triangular object in test scene A, which has been matched with its model image. Figure 5.5(a) still contains falsely matched edges (the two black lines in Figure 5.5 (a)) since we consider only vector magnitudes at this step. Figure 5.5 (b) also shows some false matches (two black lines) for the square in Figure 5.4(a). The false matches' inner angles differ from those in the model dataset. Therefore, inspecting their inner angles, and thereafter also surface invariance eliminates such false positive vectors.



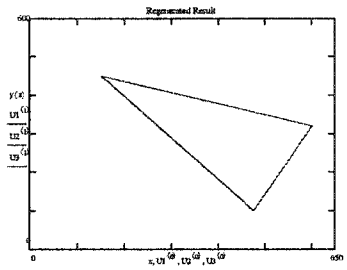
(a) Co-edge search result for triangle (b) Co-edge search results for square

Figure 5.5 Results of co-edge search

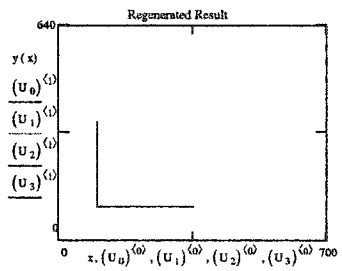
(3) Search for Loops

Since test vectors are arbitrarily drawn between interesting points without any prior knowledge, the output of their junction and loop matches still include misleading results as shown in Figure 5.5. Now we find loops from the identified co-edges (or junctions). By applying the second-level rule, we can extract loops from junctions. The key element

of the loop search is the comparison of inner angles between any extracted loop in the test scene, and loops in the model data database. The loop search result for the example of Figure 5.4 is shown in Figure 5.6.



(a) Loop search result for triangle

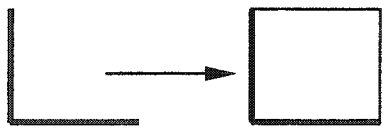


(b) Loop search result for square

Figure 5.6 Match result after loop search and surface invariance for experiment 1

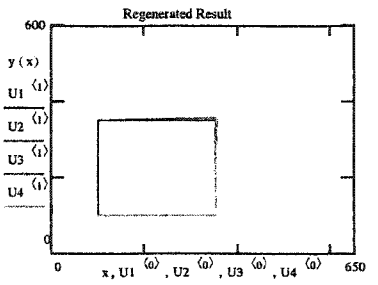
(4) Verify Surface Invariance and Evaluate DOU Value

Following the loop identification we apply the advanced level rules because false positive matches can still exist (not the case in Figure 5.6). The advanced rules confirm the match and remove false positive matches by applying the transformation invariance rule. As introduced in chapter 3, the key idea is to store the directionality of model loops. This verification step removes not only unwanted loop matches but also resolves duplicate matches from 3D shape symmetries. After completing the loop search we evaluate the degree of uncertainty (DOU) and verify whether the match is exact, over-determined, or under-determined (overlapping). Figure 5.6 (a) shows a perfect matching result for the triangle after removal of two "unwanted" co-edges by evaluating loop invariance. The DOU value for the triangle results as zero. On the other hand, Figure 5.6 (b) shows only one loop match for the square. The DOU value for the square is -3. To achieve the perfect square as total result, we initiate additional probabilistic reasoning.



Loop (Key word) Geometry (Page number)

(a) Loop matching by probabilistic reasoning



(b) Result of probabilistic reasoning

Figure 5.7 Probabilistic reasoning for square object

As shown in Figure 5.7 (a), an intact loop belongs to the square, creating a key word to the index. With probabilistic reasoning, we identify the square model object as a possible match of a hidden object. The reasoning process attempts to align model loops with the recovered loop and determine matching.

Matching model	Found co-edges	Matched loops from junction	DOU value	Scores	Decision
Triangle (Model image A)	7	3	0	3	Perfect match for triangle
Square (Model image B)	2	1	-3	1	Probabilistic reasoning for square match.

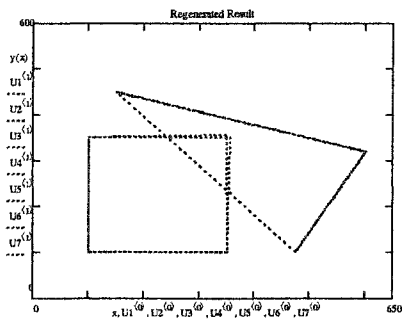
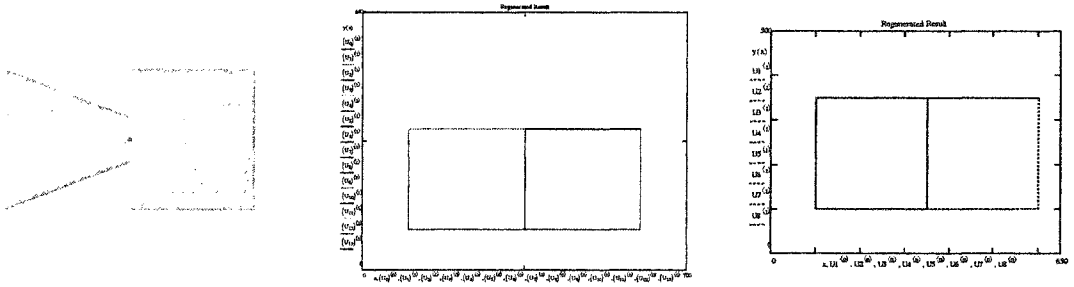


Figure 5.8 Statistics for experiment 1

The surface invariance match can verify the hypothesis. Since the model contains all four loops that define a square object and its surface invariance, we can conduct all remaining tests for invariance. After finding a correct or very close match of loop(s) the candidate object's related surfaces or its super perimeter are verified. In the example, the search terminates by identifying the occluded object as a square. Figure 5.8 shows statistics of probabilistic reasoning for experiment 1.

5.2.3 Experiment 2: Find Model Object from Test Scene B

Figure 5.9 shows recognition from test scene B in Figure 5.3 as another example. Here no match for the triangle (model image A) exists, and eight matches for the square. The latter is therefore an over determined case (DOU value is –3 for the triangle and +4 for the square). The loop search result is shown in Figure 5.9 (b). The final result is shown in Figure 5.9 (c). One can notice that it is impossible to identify the triangle since the |DOU| for the triangle is the same as the number of loops for the triangle.



(a) Test scene B (b) Loop search results (c) Final match result

Figure 5.9 Over determined problem in tests scene B

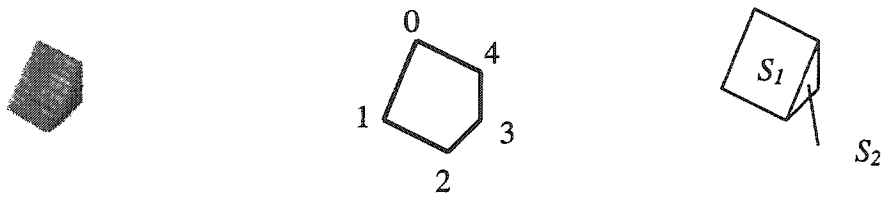
5.3 Recognition Experiments with Real 3D Objects

5.3.1 Challenges

For 3D real images, the image data from the test scene become more sensitive to errors than the synthetic images created by CAD tools. Therefore the same problems observed previously in 2D image analysis become more serious: e.g. round off errors, noise, and ambiguity. Other problems arise from projected shape variations with viewing angle, and to a lesser degree from aspects of the projected geometry such as symmetry, rotation, and scale.

5.3.2 Experiment 3: Verification of the Probabilistic interpolation in 3D

The image database of the prism in Figure 5.10 consists of contours recorded at azimuth angles of 0°, 15°, and 30° respectively. Only these three images are considered for the verification of the probabilistic interpolation for the 3D case. The super perimeter and surfaces for the prism are defined in Figure 5.10. All images of the prism have five loops for the super perimeter, four loops for surface 1, and three loops for surface 2, as well as six edges. The numbers assigned to the super perimeter (Figure 5.10b) define loop connectivity.



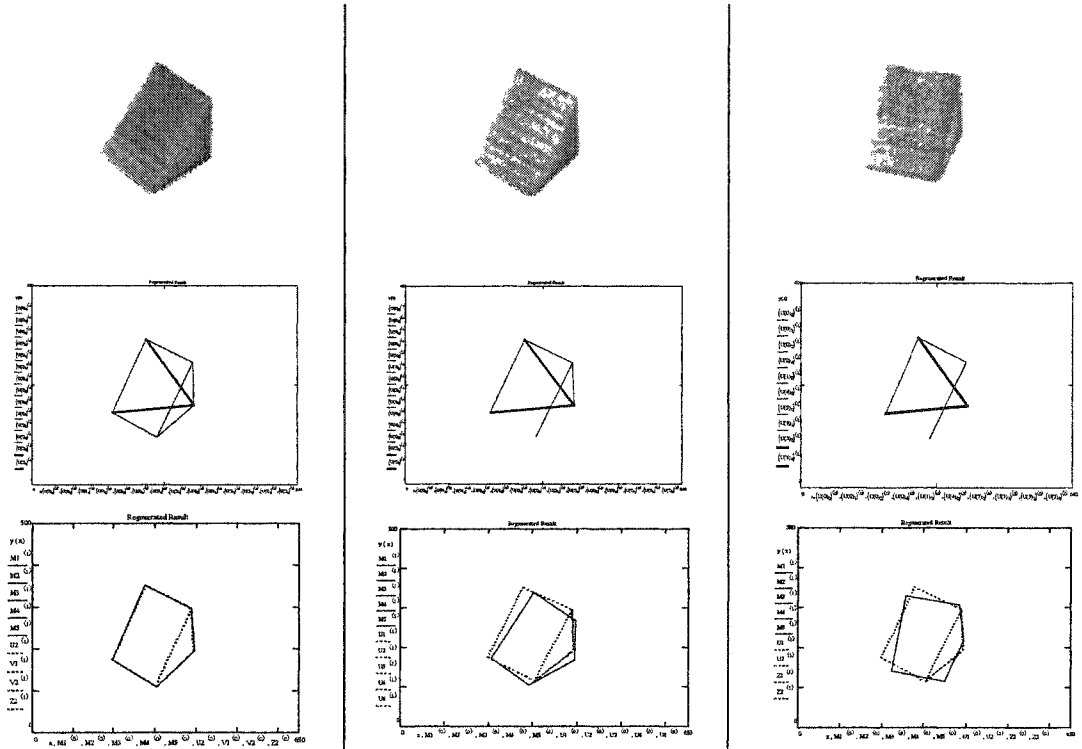
(a) Prism (b) Super perimeter (c) Surface 1 & 2

Figure 5.10 Super perimeter and surfaces of prism

Figure 5.11 shows the matching results after using the probabilistic viewing effect. In Figure 5.11, the top row shows the original image, followed by co-edge search results in the second row. The third row presents the final result after loop and surface invariant matching has removed false positive matches. The superimposed dotted image in red is the reference image contour.

(1) Perfect Match

Figure 5.11 (a) presents the case of complete agreement between model and test scene.



(a) 0 degree (b) 15 degrees (c) 30 degrees

Figure 5.11 Matching results for the same 3D object at varying viewing angles. All three views are compared to the model for view (a) (leftmost column)

Column a: Perfect match
Column b: 15 degree deviation from model viewing angle
Column b: 30 degree deviation from model viewing angle

The analysis shows not only exact matching of loops, but also additional co-edges (bold lines in second row). Analogous to 2D recognition, such false positive matches are also commonly observed in 3D real object recognition. False positive matching is due to not only to round off error and noise, but is also caused by the many possible positions the object can assume in space, and which must be considered.

Each search algorithm in 3D has been executed in parallel for each surface, as well as for the super perimeter, as discussed in chapter 4. After finding the right matches of loops using surface invariance the degree of uncertainty (DOU) is evaluated according to the definition. The search strategy follows the DOU values for search termination for the identified surfaces, and thus for the sequence in which the search proceeds.

In the present example of the perfectly matched wedge, the match score is 6 from Table 5.2 (a): since four matched loops all belong to super perimeter, we add a 50% bonus. Table 5.2 (a) shows the matching results in detail at each step of the recognition process depicted in Figure 5.11 (a).

Table 5.2 (a) Recognition statistics for perfect match example of Fig. 5.11(a)

Number of Co-edges	Number of matched Loops	DOU value	Cumulative Scores (Number of matched loops + 50% bonus)	Decision
8	12 loops	0	14.5	Perfect match

(2) Match At 15 Degree Viewing Angle Deviation

In Figure 5.11 (b) the prism has been rotated 15-degrees relative to the original position of Figure 5.11 (a). We attempt a match of the model of Figure 5.11 (a) with the rotated

test scene of Figure 5.11 (b). Now only two perfect loop matches with the super perimeter are found. Two perfect matches with the model are found for surface 1. Two matches in surface 1 also belong to the super perimeter. Due to the change of viewing angle one loop at the bottom of the prism no longer matches with the model, thus the surface 2 only one match is found. This was predicted result in Chapter 3: obtuse angled loops change more pronouncedly with varying viewing angles compared to acute angles. Referring to Table 5.2(b) the super perimeter match has the highest score. The algorithm first attempts to recover the super perimeter, and then evaluates other surfaces. Since the super perimeter bounds surfaces, this sequence of reasoning constitutes backward processing compared to the normal forward search sequence discussed in Chapter 4.

Table 5.2 (b) Recognition statistics for match example of Fig. 5.11(b) at 15 degrees viewing angle deviation

Co-edges	Loops	DOU value	Scores	Decisions
	2 for Super perimeter	-3	3	Still has the highest credit
6	2	-2	2.5	One of two loop belongs not only surface 1 but also super perimeter so we gave extra credit of 50 percent
	1	-2	1	Possible to recover but the lowest credibility

(3) Match At 30 Degree Viewing Angle Deviation

Figure 5.11 (c) summarizes the reasoning process for match between prisms at 30-degree difference. Another edge in surface 2 no longer matches the model of Figure 5.11 (a).

The super perimeter has one perfect match, and there are two perfect matches for surface 1. There is no match for surface 2. Therefore unlike in the preceding experiment, the algorithm attempts to find a match using surface 1. The permissibility of matching surface 1 with the model depends on the user defined maximum deviation between model and test scene. The selection of allowable deviation levels between model and test scene influences the search process directly with regard to the number of required comparisons and thus the total time required for completion of the search. The allowable deviation levels in this example result in a DOU value that still holds the possibility for recovery, albeit at a lower probability for success than the preceding experiments.

Table 5.2 (c) Recognition statistics for match example of Fig. 5.11(c) at 30 degrees viewing angle deviation

Co-edges	Loops	DOU value	Scores	Decisions
5	1 from super perimeter	-4	1.5	Possible to recover but lower credit than surface 1
	2 from surface 1	-2	2.5	One of two loop belongs not only surface 1 but also super perimeter we gave extra credit of 50 percent
	0	-3	0	Impossible to recover

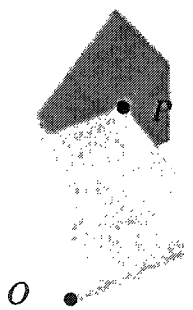
5.3.3 Experiment 4: Locating a Model Object in Test Scene C

This experiment performs 3D object recognition from a real image containing multiple objects. As always, there is no prior knowledge from the test scene except point

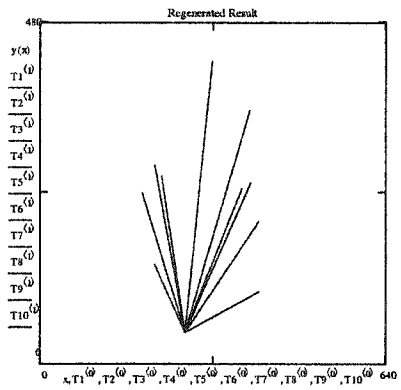
coordinates. The procedure up to the loop search is the same as in the 2D case. Then, after evaluating the DOU values, a super perimeter search as well as surface searches is performed. As before, the results are ranked in terms of scores assigned to each hypothesis.

(1) Generating the Test Vector

Figure 5.12 (a) shows test scene C with a partially occluded prism. The test vector set is the same as for 2D synthetic object recognition. The test scene contains are a total of 55 test vectors. Figure 5.12 (b) shows one set, drawn from point *O*. The results from corner detection show that it is almost impossible to detect point *P*. The overlapping prism creates new deep convex or concave junctions, which become false interesting points.



(a) Test scene C



(b) Test vectors from point *O*

Figure 5.12 One of eleven test vector sets from test scene C at point *O*

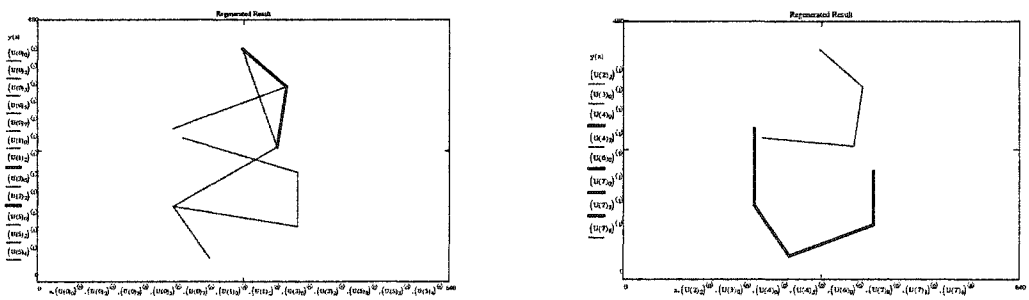
(2) Finding Co-edges from Test Vector Sets

Figure 5.13 shows the result of the co-edge search. Figure 5.13 (a) contains many complicated co-edges after the match between the prism and test scene C because the

overlapping objects created new corner points and removed some original ones. The resulting ambiguity causes many false positive results even if all objects in the test scene belong to the model image database. Figure 5.13 (b) shows the co-edge search results after matching the cube in the test scene with the model, producing better results. However, even with the cube located in front of the prism, the corner detection algorithm lacks point *P* in Figure 5.12 (a) on the overlapping part of prism.

(3) Loop search

The loop search originates from the detected co-edges. The bold lines in Figure 5.13 (a) and (b) show one loop match for the prism (bold line) and three loop matches for the cube, respectively. Therefore the DOU value for the cube search in the test scene C results as under-determined (possibly overlapped). Probabilistic reasoning is applied to both candidate objects. Overlapping affects not only the occluded object, but also the one in front since both objects lost interesting points.



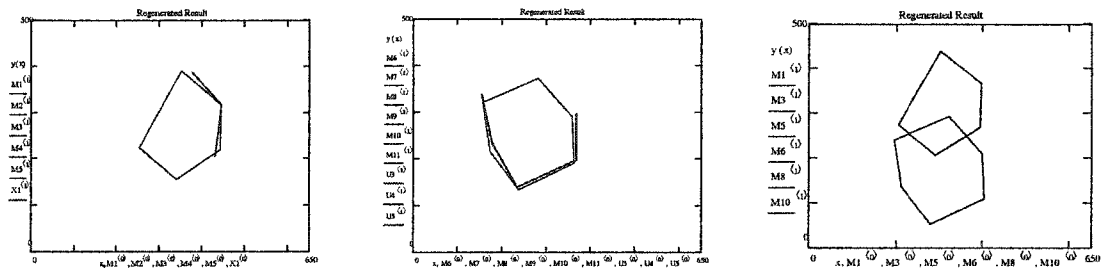
(a) Co-edge with prism (b) Co-edge with cube

Figure 5.13 Result of co-edge search from test scene C

(4) Verify the Invariant Vectors

The next step is to find the nearest neighbors of existing intact loops from the model database. Since the cube has more intact loops, and the detected loops in the cube all

belong to super perimeter, the algorithm then attempts to recover the super perimeter, and then to collect the surfaces, which belong to that super perimeter. For the prism, the only matching loop *luckily* also belongs to the super perimeter so the algorithm follows the same procedure as for the cube. Figure 5.14 (c) shows the final matches with depictions picture of the two identified model images, which were each identified from nearest neighbor searches. Table 5.3 summarizes the decision functions and the rules applied to verify the hypotheses.



(a) Partial match for prism (b) Partial match for cube (c) Final match result

Figure 5.14 Final match after loop search and surface invariance

The recovered geometry in Figure 5.14 (c) contains not the test scene objects, but they're nearest neighbor model images found during the match process. With the exception of the occasional perfect match the final match deviates from the test scene since the 3D model images are matched by application of the probabilistic interpolation within the bounds of acceptable error margins. It is possible to obtain several matches for the same test scene, but such cases are not discussed here. The frequency of multiple matches for a single scene depends on the number of model images in the model database: The smaller the interval between stored views, the fewer multiple matches will

result, as discussed in Chapter 3. Smaller intervals between views result in larger databases and larger numbers of iterations. As discussed in Chapter 3 a 15-degree interval between model images constitutes a reasonable compromise. The next experiment demonstrates a case of failure since the test scene contains no related images in the model database.

Table 5.3 Recognition results from various stages of matching algorithm for Experiment 4 (image containing two overlapping objects)

THE RULE GROUPS			TIMES FIRED	NUMBERS OF MATCHES	ACTIONS OF THE RULES
Low level rule	Prism		55*6	12	Find co-edges
	Cube		55*9	8	
Intermediate level rule	Prism	Super	12*5	5	Find loops from co-edges
		S1	12*4	2	
		S2	12*3	1	
	Cube	Super	8*6	3	
		S1	8*4	2	
		S2	8*4	0	
		S3	8*4	1	
Advanced level rule	Prism	Super	5*4!	1	Verify invariance vector from matched loops
		S1	2*3!	0	
		S2	1*2!	1	
	Cube	Super	3*6!	3	
		S1	2*4!	0	
		S2	0*4!	0	
		S3	1*4!	0	

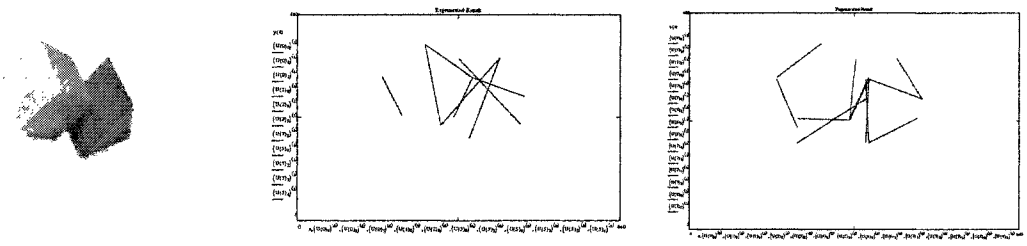
5.3.4 Experiment 5: Locating a Model Object from Test Scene D

In test scene D, the cube leans on the prism, and the photo was taken from a different elevation angle when compared to test scene C. Since there is difference if viewing angle changes in the test scene, then the scene become totally unknown object to model

database. And we try to demonstrate how it is different when unknown object comes out in the test scene.

(1) Search for Loops and Evaluate DOU Values

Since generating test vectors and finding co-edges follows the same procedures as in the previous sections we move immediately to the co-edge search results, depicted in Figure 5.15. Figures 5.15 (b) and (c) show the co-edge search result for the prism and cube, respectively.



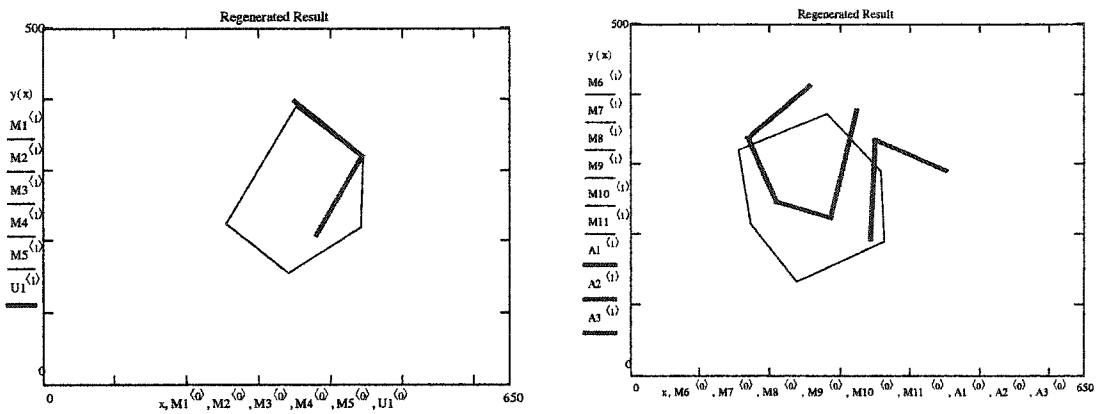
(a) Test scene D (b) Co-edges for prism (c) Co-edges for cube

Figure 5.15 Recognition results for test scene D

Once again, the overlapping objects result in a loss of interesting points compared to the model objects, as well as in the creation of additional points not belonging to the data base. Thus the co-edges are rather complicated. Correct matches are still possible, even if the viewing angles of an object in the test scene deviate considerably from the stored data sets in the model database. The likelihood for correct recognition deceases as the difference between test scene and stored views increases. The prism has one matched loop (bold line) in Figure 5.16 (a) the cube shows three-matched loops (bold lines) in Figure 5.16 (b).

(2) Parallel Search for Super Perimeter and Surface

To search for the prism in the test scene, we conduct three steps in parallel, (i) find match with super perimeter, (ii) find match with surface 1, and (iii) find match with surface 2. The algorithm finds no match from Figures 5.16 (a) and (b). The matching loop in Figure 5.16(a) suggests that the prism could be correctly identified. However, the edge length is too short and therefore not within the acceptable bounds of tolerance. In Figure 5.16 (b), clearly no surface invariance between the three matched loops exists. Thus, even though loop matches exist, one cannot conclude matching without verification of surface invariance by evaluating the transformation vectors. The algorithm concludes that no match exists in test scene D concerning the model image database.



(a) Matching result for prism (b) Matching result for cube

Figure 5.16 Final match after loop search and surface invariance

5.4 Summary

Rule-based pattern matching produces satisfactory recognition results for 2D synthetic images. For this image category, the model image base is relatively free of noise and

other errors. Object recognition is generally accurate even in the case of overlapping and partial occlusion. For the 3D real objects, the levels of ambiguity of data and errors accumulated by probabilistic interpolation decreases the reliability of matching and particularly for obtuse-angled loops. Pattern matching for scenes containing overlapping objects depends strongly on the objects' relative positions with regard to the algorithm's capability to identify valid loops in the scene.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

The objective of this dissertation is the development and validation of automated methods for machine-based object recognition. Emphasis was placed on search methodologies that minimize the database size requirements, as well as on robust and noise-tolerant identification methods that ensure correct recognition at high levels of reliability. Strategies to minimize processing time include a database organized for feature-based indexing, nearest neighbor searches, and iterative search path optimization based on the current state of information. Validation of search results is performed at multiple intermediate steps during the search both to increase accuracy as well as to prune the search space and reduce execution time. A database of 3D object models was created, in which object contours are reduced to interesting points and their spatial relationships to the entire object. Models were extracted from 2D gray-scale digital images. The database search and validation algorithm is configured as a rule-based recognition system, which employs probabilistic interpolation and reasoning.

The search algorithm reaches its conclusions based on four main elements:

The indexing key and the feature-based geometrical model and probabilistic interpolation and the rule-based reasoning procedure. They designed to improve speed and reliability during the search. Details of these four elements follow below:

1. **Indexing key**: The results in the Chapter 5 support the claims that indexing is a powerful tool and that multiple representations provide robustness. These properties allow the search algorithm to function efficiently in the presence of moderate noise levels and occlusion.
2. **Feature-based indexing**: Feature-based indexing reduces the size of the hash table since the database consists of loops and their geometric invariance only.
3. **Probabilistic interpolation**: Similarity analysis of neighboring 3D images was performed and the accuracy and reliability have been found satisfactory.
4. **Rule-based reasoning**: Knowledge-based system was introduced to recognize object without orderly manner of vertex coordinates from test scene and regenerate the test scene as recognition result. The definition of degree of uncertainty (DOU) played two important roles. One is determine if there is occlusion or missing loops and the other one is determine the termination of iteration.

6.2 Limitations of the Algorithm

1. **Special cases of model views**: The expert system fails in those cases where only one surface of a 3D object is visible. In this situation, insufficient information exists for a decision about the nature of the scene (plane or 3D).

2. **Incomplete search**: Because the 3D search is probabilistic, it is always possible that the indexing algorithm may fail. The likelihood of correct recognition increases in proportion with the number of feature groupings, with each group increasing the probability for achieving correct matches.

6.3 Recommendations

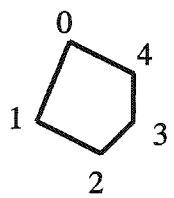
1. **Probability estimation**: The accuracy of the probability estimates used for ranking could be improved by adding additional distinctive features to the database.
2. **Feature data grouping**: improving the techniques of data grouping will be important for increasing the robustness of the system and deceasing database size, as well as minimize iteration.
3. **Verification**: Both the probabilistic reasoning process and the set of recognition rules can be further refined. Especially after the loop estimation has been stabilized, further refinement of rules could enhance the recognition accuracy.
4. **Process Automation:** While this effort was focused on developing a proof of concept, both the data base creation process and the pattern matching can be automated and integrated.

APPENDIX

A.1 Sample Data in Off-line Processing

A.1.1 Edge and Loop Data Sample

Figure A.1.1 below shows format of loop and edge data for super perimeter.



(a) Super perimeter

$$\text{Super} := \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 0 \\ 2 & 3 & 4 & 0 & 1 \\ 1.711 & 1.919 & 2.154 & 2.194 & 1.446 \\ 192.3 & 109.64 & 119.89 & 99.05 & 118.41 \\ 109.64 & 119.89 & 99.05 & 118.41 & 192.3 \end{pmatrix}$$

(b) Matrix data for super perimeter

Figure A.1.1 Example for super perimeter database for prism model

The first three rows in the super matrix at Figure A.1.1 (b), shows connectivity of five loops consist super perimeter at Figure A.1.1 (a). And the fourth row in the super matrix represents inner angle value in radian. And the last two rows indicate magnitude of two edges belong to each loop at pixel length scale.

A.1.2 Surface invariance

The example for the invariance vector between loop 012 and loop 340 is shown in Figure A.1.2. And the mathematical form is in equation (A.1).

$$<< Loop012, Loop340, \angle\theta_1, \angle\theta_2, \overline{R} >> \qquad (A.1)$$

Here θ_1 and θ_2 is directionality between loop 012 and 340 and R is the magnitude of vector v.

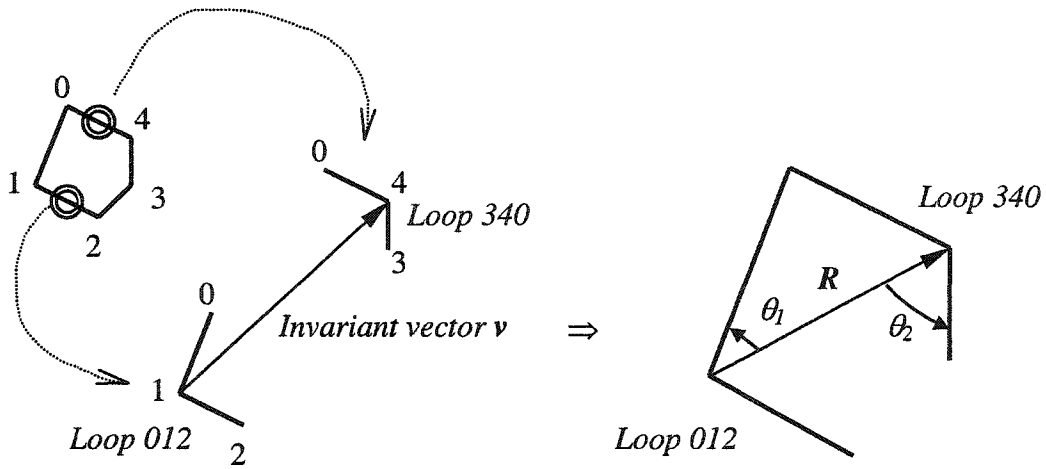


Figure A.1.2 Details of surface invariance between loop 012 and 340

To verify their invariance, we used translation and rotation matrix as shown Figure 3.3 in chapter 3. During the recognition we verify hypothesis using transformation invariance. In this stage, θ becomes rotation angle and R becomes translation distance. As a matter fact, it is exhaustive progress to build those invariance data along with edge and loop data as well.

A.2 Table of 3D model images for prism

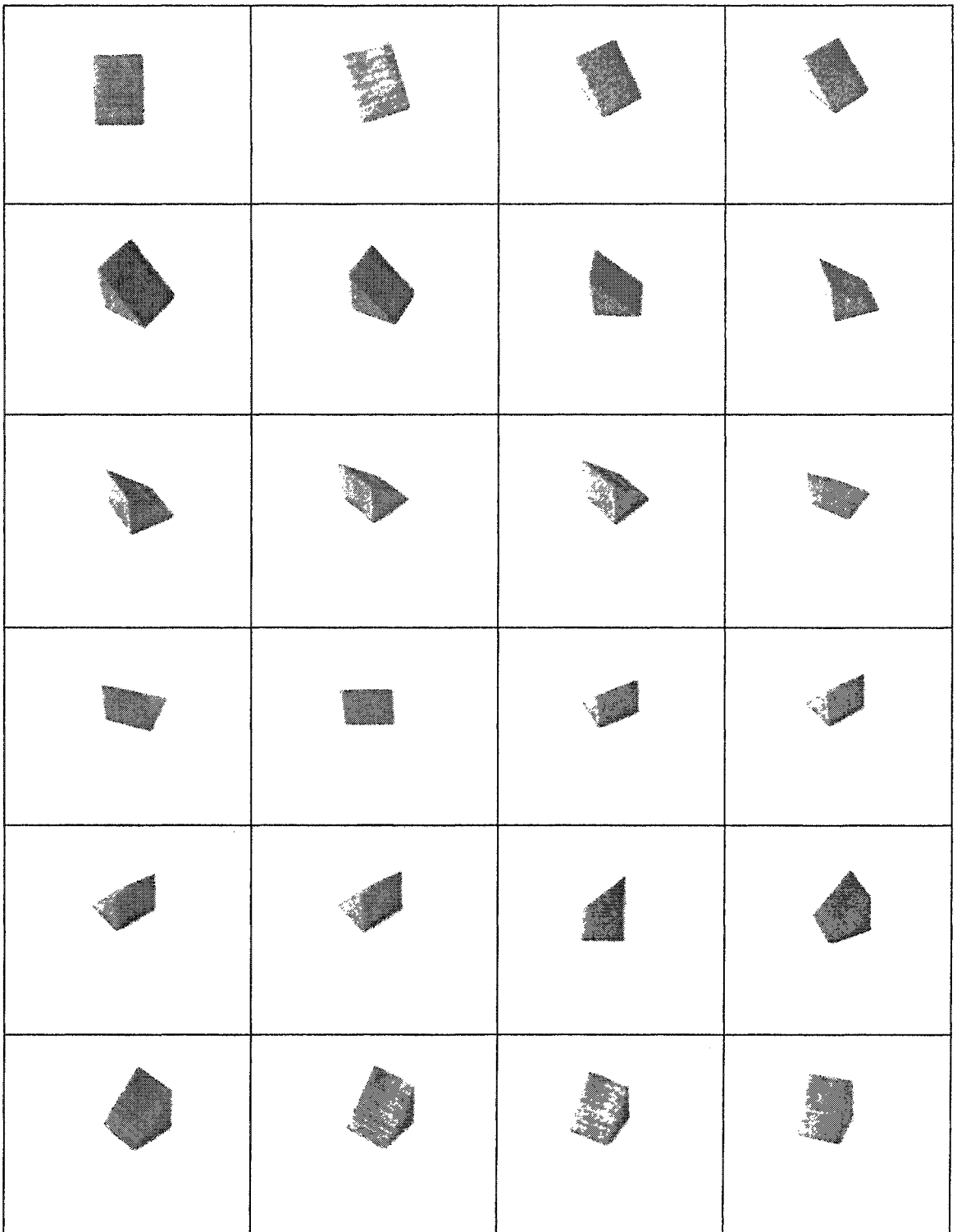


Figure A.2.1 List of images for prism database taken at 15-degree interval. (Elevation angle is fixed)

A.3 Sample CLIPS code

```
;*****************************************************************
;
;       Clips Example for Feature-Based Indexing
;
;*****************************************************************
(deffacts m_data

;;edge data

        (m_data edge1 250 0)
        (m_data edge2 250 90)
        (m_data edge3 250 180)
        (m_data edge4 250 270)

;;loop data

        (m_data loop1 012 90)
        (m_data loop2 123 90)
        (m_data loop3 230 90)
        (m_data loop4 301 90))

;;set test scene data

(deftemplate group
        (slot name)
        (slot connectivity)
        (slot mag))

;set eight rules to find match
;

(defrule find-match-edge1
        (m_data edge1 ?x ?y)
        (group (name ?name) (mag ?x))
=>(printout t ?name "    " crlf))

(defrule find-match-edge2
        (m_data edge2 ?x ?y)
        (group (name ?name) (mag ?x))
=>(printout t ?name "    " crlf))

(defrule find-match-edge3
        (m_data edge3 ?x ?y)
        (group (name ?name) (mag ?x))
```

```
=>(printout t ?name "    " crlf))

(defrule find-match-edge4
        (m_data edge4 ?x ?y)
        (group (name ?name) (mag ?x))
=>(printout t ?name "    " crlf))

(defrule find-match-loop1
        (m_data loop1 ?x ?y)
        (group (connectivity ?connectivity) (mag ?y))
=> (printout t ?connectivity "    " crlf))

(defrule find-match-loop2
        (m_data loop2 ?x ?y)
        (group (connectivity ?connectivity) (mag ?y))
=> (printout t ?connectivity "    " crlf))

(defrule find-match-loop3
        (m_data loop3 ?x ?y)
        (group (connectivity ?connectivity) (mag ?y))
=> (printout t ?connectivity "    " crlf))

(defrule find-match-loop4
        (m_data loop4 ?x ?y)
        (group (connectivity ?connectivity) (mag ?y))
=> (printout t ?connectivity "    " crlf))

(deffacts test_vector

;;;************** test vector data ******************

        (group (name t0_0)(mag 0))
        (group (name t0_1)(mag 250))
        (group (name t0_2)(mag 250))
        (group (name t0_3)(mag 375))
        (group (name t0_4)(mag 546))
        (group (name t0_5)(mag 291))
        (group (name t0_6)(mag 291))
        (group (name t0_7)(mag 353))

        (group (name t1_0)(mag 250))
        (group (name t1_1)(mag 0))
        (group (name t1_2)(mag 353))
        (group (name t1_3)(mag 125))
        (group (name t1_4)(mag 333))
        (group (name t1_5)(mag 150))
```

(group (name t1_6)(mag 269))
(group (name t1_7)(mag 403))

(group (name t2_0)(mag 250))
(group (name t2_1)(mag 355))
(group (name t2_2)(mag 0))
(group (name t2_3)(mag 450))
(group (name t2_4)(mag 500))
(group (name t2_5)(mag 269))
(group (name t2_6)(mag 150))
(group (name t2_7)(mag 111))

(group (name t3_0)(mag 375))
(group (name t3_1)(mag 125))
(group (name t3_2)(mag 450))
(group (name t3_3)(mag 0))
(group (name t3_4)(mag 253))
(group (name t3_5)(mag 195))
(group (name t3_6)(mag 336))
(group (name t3_7)(mag 477))

(group (name t4_0)(mag 546))
(group (name t4_1)(mag 333))
(group (name t4_2)(mag 500))
(group (name t4_3)(mag 253))
(group (name t4_4)(mag 0))
(group (name t4_5)(mag 259))
(group (name t4_6)(mag 351))
(group (name t4_7)(mag 468))

(group (name t5_0)(mag 291))
(group (name t5_1)(mag 150))
(group (name t5_2)(mag 269))
(group (name t5_3)(mag 195))
(group (name t5_4)(mag 259))
(group (name t5_5)(mag 0))
(group (name t5_6)(mag 141))
(group (name t5_7)(mag 282))

(group (name t6_0)(mag 291))
(group (name t6_1)(mag 269))
(group (name t6_2)(mag 150))
(group (name t6_3)(mag 336))
(group (name t6_4)(mag 351))
(group (name t6_5)(mag 141))
(group (name t6_6)(mag 0))

```
(group (name t6_7)(mag 141))

(group (name t7_0)(mag 353))
(group (name t7_1)(mag 403))
(group (name t7_2)(mag 111))
(group (name t7_3)(mag 477))
(group (name t7_4)(mag 468))
(group (name t7_5)(mag 282))
(group (name t7_6)(mag 141))
(group (name t7_7)(mag 0))


;;;********** co-edge data **********

(group (name loop_1)(connectivity 132)(mag 90))
(group (name loop_2)(connectivity 023)(mag 90))
(group (name loop_3)(connectivity 013)(mag 90))
(group (name loop_4)(connectivity 102)(mag 90)))
```

BIBLIOGRAPHY

Aggarwal, J. K., Ghosh, J., Nair, D., and Taha, I; "A Cpmparative Study of Three Paradigms for Object Recognition – Baysian Statistics, Neural Networks and Expert Systems", Image Understanding, A Festschrift for Axriel Rosenfeld, pp. 241 – 262, 1996.

Basri, Ronen; Ullman, Shimon; "The Alignment of Objects with Smooth Surfaces" CVGIP: Image Understanding, Vol. 57, No. 3, pp. 331-345, May 1993.

Bataille, M. C. and Robert Galley (1997) "L'aval du cycle nucléaire " Rapport de l'Office Parlementaire d'Evaluation des Choix Scientifiques et Technologiques No 612, http://www.senat.fr/rap/o97-612/o97-612.html

Bebis, George; Georgiopoulos, Michael; Shah, Mubarak; Vitoria Lobo, Niels; "Indexing Based on Algebraic Functions of Views" Computer Vision and Image Understanding, Vol. 72, No. 3, December, pp. 360-378, 1998.

Beis, Jeffrey S.; Lowe, David G.; "Indexing without Invariant in 3D Object Recognition" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 10, pp. 1000-1015, October 1999.

Ben-Arie, Jezekiel; "The probabilistic Peaking Effect of Viewed Angles and Distances with Application to 3-D Object Recognition" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12, No. 8, pp. 760-774, 1990.

Biederman, I; "Human image understanding: recent research and a theory", CVGIP, Vol. 32, pp. 29–73, 1985.

Boidron M. , N. Chauvin, J.C. Garnier , S Pillon , G. Vambenepe (2000) " Transmutation Studies In France, R&D Programme On Fuels And Targets," Proc. Conf. on Partitioning and Transmutation, Madrid, Dec.
http://www.nea.fr/html/pt/docs/iem/madrid00/Proceedings/Paper75.pdf

Burns, J. Brian; Weiss, Richard S.; Riseman, Edward M.; "View Variation of Point-Set and Line-Segment Features" IEEE Transactions on Parretn Analysis and Machine Intelligence, Vol. 15, No. 1, pp. 51-68, January 1993.

Califano, Andrea; Mohan, Rakesh; "Multidimensional Indexing for Recognizing Visual Shapes" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 4, pp. 373-392, April 1994.

Canny, J.; "A Computational Approach to Edge Detection" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679-697, November 1986.

Chin, Ronald T.; Dyer, Charles R.; "Model–Based Recognition in Robot Vision" Computing Surveys, Vol. 18, No. 1, pp. 67-103, March 1986.

Clemens, David T.; Jacobs, David W.; "Space and Time Bounds on Indexing 3-D Models from 2-D Images" IEEE Transactions on Pattern analysis and Machine Intelligence, Vol. 13, No. 10, pp. 1007-1017, October 1991.
CLIPS, [http://www.ghg.net/clips/].

Duda, R.O., Hart, P. E; "Pattern Classification and scene Analysis", New York, Wiley, 1973.

Giarrantano, J.; "Expert Systems: Principles and Programming", Brooks Cole, 1998.

Grimson, W. E. L.; Huttenlocher, D. P.; "On the Sensitivity of Geometric Hashing" Proceeding on International Conference of Computer Vision, pp. 334-338, 1990.

Grimson, W. E. L.; Huttenlocher, D. P.; T. D. Alter; "Recognizing 3D Objects from 2D Images: An Error Analysis" Proceeding on International Conference of Computer Vision, pp. 316-321, 1992.

Guzman, A.; "Decomposition of a visual scene into 3-D bodies", Academic Press, New York, 1969.

Haas, D., J. Somers, A. Renard, A. La Fuente (1998) "Feasibility of the Fabrication of Americium Targets"
Proceedings Fifth OECD/NEA Information Exchange Meeting on Actinide and Fission Product Partitioning and Transmutation," Mol, Belgium, Nov.
http://www.nea.fr/html/pt/docs/iem/mol98/session3/SIIIpaper1.pdf

Huttenlocher, Daniel P.; Ullman, Shimon; "Recognizing Solid Object by Alignment with Image" International Journal of Computer Vision, Vol. 5, No.2, pp. 195-212, 1990.

Lamdan, Yehezkel; Schwartz, Jacob T.; Wolfson, Haim J.; "Affine Invariant Model-Based Object Recognition" IEEE Transactions on Robotics and Automation, Vol. 6, No. 5, pp. 578-589, October 1990.

Lee, C; Slagle, James R.; "An experimental study of an object recognition system that learns", Pattern Recognition, Vol. 27, No. 1, pp. 65-89, 1994.

Lee, J; Mauer, G.; "Non-Contact Target Acquisition and Object Identification for Robotic Grasping", Proc. ISCA 13th Int. Conf., pp. 592-597, 2000.

Lowe, David G.; "Three-Dimensional Object Recognition from Single Two-Dimensional Images", Artificial Intelligence, 31 pp. 355-395, 1987.

Magnor, M.; "Geometry-Based Automatic Object Localization and 3-D Pose Detection", IEEE Proceeding, Image Analysis and Interpretation, Santa Fe, April 2002.

Mauer, G; Renno, J.; "Design and Virtual Testing of Robotic Assembly Processes for Hot Cells", 10th International Topical Meeting on Robotics and Remote Systems for Hazardous Environments, University of Florida, April, 2004.

Medioni, G. and Yasumoto, Y.; "Corner Detection and Curve Representation Using Cubic B-Splines" Computer vision, Graphics, and Image Processing, 39, 267-278, 1987.

Meyer, M. (2001) "The U.S. Program for the Development of Inert-matrix Fuel for Transmutation Systems," Presentation to the AAA project, Las Vegas, NV, June http:// aaa.nevada.edu/pdffiles/Meyer6_21_01.pdf

NEA (1999) "Status and Assessment Report on Actinide and Fission Product Partitioning and Transmutation," http://www.nea.fr/html/pt/docs/1999/neastatus99/Phase1_report.html and http://www.nea.fr/html/pt/docs/1999/neastatus99/AnnexF.pdf (Cost Assessment) and http://www.nea.fr/html/pt/docs/1999/neastatus99/Part1.pdf (P&T Technologies)

NEA (2000) "Status and Assessment Report on Actinide and Fission Product Partitioning and Transmutation," http://www.nea.fr/html/ndd/docs/2001/nea3108-actinide.pdf, December.

Olson, Clark F.; "Probabilistic Indexing for Object Recognition" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 5, pp. 518-522, May 1995.

Roberts, L. G.; "Machine Perception of 3-D solids", MIT Press, Cambridge, MA, 1965.

Shapiro, L. G; Moriarty, D.; Haralick, R. M.; " Matching 3D objects using a relational paradigm", Pattern Recognition, 17(4), pp. 385 – 405, 1984.

Shashua, Amnon; "Algebraic Functions for Recognition" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 8, pp. 779-789, August 1995.

Shimshoni, I; Ponce, J; "Probabilistic 3D Object Recognition", International Journal of Computer Vision, 36(1), pp. 51-70, 2000.

Stein, F.; Medioni, G.; "Structural Indexing: Efficient 3D Object Recognition" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, pp. 125-145, February 1992. (a)

Stein, F.; Medioni, G.; "Structural Indexing: Efficient 2D Object Recognition" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 12, pp. 1198-1204, December 1992. (b)

Ullman, Shimon; Basri, Ronen; "Recognition by Linear Combinations of Models" IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 10, pp. 992-1006, October 1991.

Weinshall, D; "Model-Based Invariants for 3-D Vision", International Journal of Computer Vision, Vol. 10, No. 1, 27-42, 1993.

Weiss, I; Ray, M; "Model-Based Recognition of 3D Objects from Single Images", IEEE PAMI, Vol. 23, No. 2, Feb. 2001.

VITA

Graduate College
University of Nevada, Las Vegas

Jae-Kyu Lee

Local Address:
1600 East University Avenue #212
Las Vegas NV 89119

Home Address:
728 Kinau Street #605
Honolulu, HI 96813

Degrees:

Bachelor of Science, Mechanical Engineering, 1988
Kon-Kuk University, South Korea

Master of Science, Mechanical Engineering, 1990
Kon-Kuk University, South Korea

Master of Science, Mechanical Engineering, 1998
University of Florida, Gainesville, Florida

Publications:

SAE 931085, 1993, "An Experiment and Performance Simulation of Diesel Engine using Artificial Intake"

Proc. ISCA 13th Int. Conf., pp. 592-597, 2000, "Non-Contact Target Acquisition and Object Identification for Robotic Grasping"

Zeitschrift fur Metallkunde Vol. 94, No. 9, 2003,"Development of SMD 32.768 kHz tuning fork-type crystals"

Plating and Surface Finishing, Vol. 9, 2003,"Effect of Post-detachment Cleaning upon the Hydrophilic Nature of Nickel Thin Film Nozzle Plates for Piezo Ink-Jet Printer Heads"

Dissertation/Thesis Title: Three Dimensional Pattern Recognition using Feature-Based Indexing and Rule-Based Search

Dissertation/Thesis Examination Committee:

Chairperson, Dr. Georg Mauer, Ph. D.
Committee Member, Dr. Mitchell Meyer, Ph. D.
Committee Member, Dr. Mohamed Trabia, Ph. D.
Committee Member, Dr. Zhiyong Wang, Ph. D.
Committee Member, Dr. Evangelos Yfantis. Ph. D.
Committee Member, Dr. Woosoon Yim, Ph. D.