EXTRACTING COMPLETE 3-DIMENSIONAL BOUNDARY REPRESENTATION FROM MULTIPLE RANGE IMAGES

by

Zukang Xu

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in the School

of

Engineering Science

Simon Fraser University

June, 1992

APPROVAL

NAME: Zukang Xu

DEGREE: Master of Applied Science (Engineering Science)

TITLE OF THESIS: Extracting Complete 3-dimensional Boundary Representation from Multiple Range Images

EXAMINING COMMITTEE:

Chairman: Dr. John Jones

Dr. Kamal K. Gupta
Senior Supervisor

Dr. Jacques Vaisey
Supervisor

Dr. Ze-Nian Li
Supervisor

Dr. Tom Calvert
Examiner

DATE APPROVED: August 10, 1992

PARTIAL COPYRIGHT LICENSE

Title of Thesis/Project/Extended Essay

Extracting Complete 3-dimensional Boundary Representation from Multiple Range Images.

Author:

(signature)

Xu, Zukang

(name)

August 10, 1992

(date)

Abstract

An important problem in computer vision is that of generating a 3-dimensional description of objects from sensed data. In robotics applications such as grasping and planning collision-free motion, symbolic 3-dimensional descriptions of objects must be available. Such 3-dimensional descriptions require explicit geometric and topological relations between faces edges and vertices, and are very similar to boundary representation (*b-rep*) used in solid modeling.

In this thesis, we present an approach to extract a complete *b-rep* description of a polyhedral object from range images taken from multiple view-points. Our system, starting from basic face models of visible surfaces of objects in each local view, extracts features, matches these features, generates rigid-body transformations that relate the local views, and finally merges these local views into a complete 3-dimensional *b-rep* description of the object. In our matching algorithm, a *triple branch structure* is defined as the main feature to determine correspondence. A coarse-to-fine searching strategy and a prioritizing procedure are designed to reduce the search time even further. The *b-rep* model is incrementally updated by merging each of the local views in a global description. A convenient and effective termination criterion is designed to monitor the integration process.

We have implemented our system in C, running on a SUN/SPARC. A synthetic range image generator has also been implemented. The system has been tested on several synthetic range images. Our system in conjunction with the synthetic range image generator also has applications in the area of robotics, *CAD* and geometric modeling.

To my mother

and

in memory of my father

Acknowledgements

My sincere thanks to my senior supervisor Dr. Kamal K. Gupta for his guidance and encouragement in the course of this project. I am very grateful to my committee members, Dr. Tom Calvert, Dr. Ze-Nian Li and Dr. Jacques Vaisey, for their assistance and helpful hints. In addition, I was lucky enough to obtain useful help from other individuals. Dr. Edgar Pechlaner gave me very useful suggestions for solving the geometry problems. Ms. Michele Valiquette helped me in revising the thesis. I would also like to express my appreciation to my colleagues, Frank tong and Louis Brassard, for their valuable discussions during this research. Many thanks to my friends for their encouragement and assistance.

Finally, very special thanks go to my wife and my son for their support and patience.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Motivation

A challenging problem for computer vision is that of generating a 3-dimensional description of an unknown scene from sensed data. As important components in model-based recognition, model-based inspection, autonomous navigation, and path planning, scene description and reconstruction have received increasing attention.

In robotics applications, it is interesting not only to recognize objects in the scene, but also to estimate as accurately as possible their position and orientation. In both cases symbolic descriptions of objects must be available. To interact with its environment, a robot has to build a description of the objects around it. For instance, to plan a collision-free path, one needs a geometrical description of obstacles [20]. To find a grasp position on a polyhedral object, one needs to reason about the relationship among the faces of the object, e.g., which two faces are parallel, etc.

[10]. Therefore, a complete 3-dimensional symbolic description of the objects is necessary.

In recent years, digitized range data has become available from both active and passive sensors, and the quality of these data has been steadily improving. Not only are depth relationships between depth map regions explicit, but the 3-dimensional shape of depth map regions approximates the 3-dimensional shape of the corresponding object surfaces in the field of view. Therefore, the process of describing and reconstructing objects by their shape should be less difficult in range images than in intensity images. That is why we have used range images. However, it is impossible to view the whole object from a single viewpoint. Particularly for a range scanner (see Section 1.4.1 for details), the occlusion problem may be severe. In order to build a complete description of the whole object, multiple views are needed.

1.2 The Problem

Our ultimate aim is to provide a range-image-based vision system for a robotic work cell to build a complete 3-dimensional geometric description of the objects in the robot's environment. This description may then be used by robotic manipulation algorithms such as collision-free path planning or grasp planning. The boundary representation (*b-rep*) is chosen to describe objects, which consists of a list of primitives (vertices, edges, faces, etc.) of objects and the geometric and topological relationships between these primitives. In particular, the desired description in our applications is very similar to the winged-edge representation — a widely used *b-rep* (see Subsection 1.4.2 for details). The specific problem considered in this thesis is

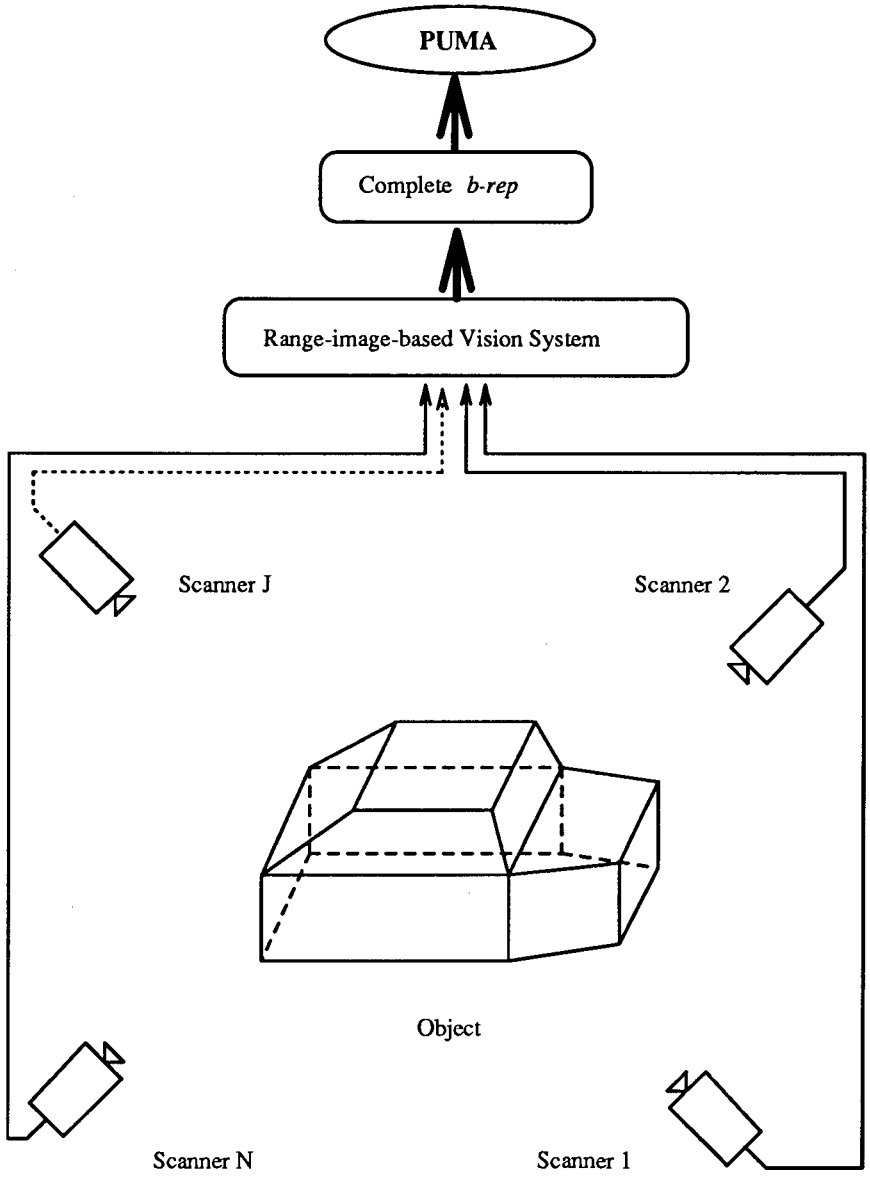


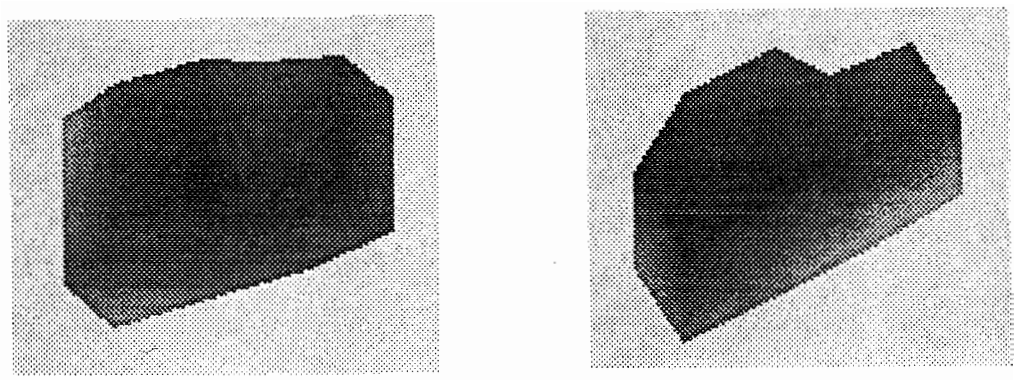
Figure 1.1: A range-image-based vision system in a robotic work cell.

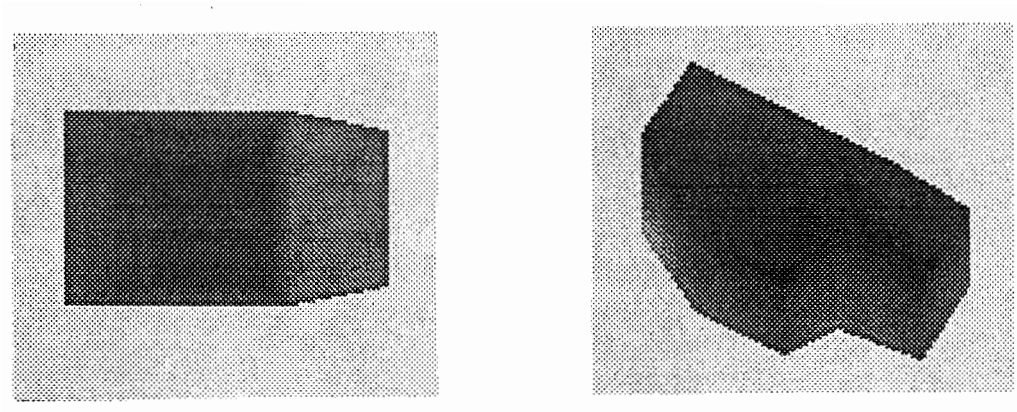
how to completely describe the geometry of an object from multiple range images.

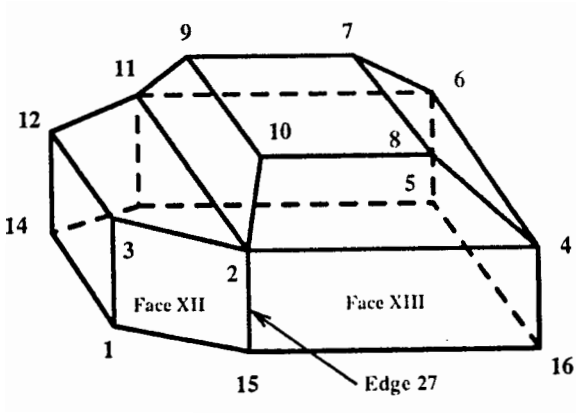
For simplicity, the sensed object is assumed to be a single, bounded, closed, and regular polyhedral object [45]. The objects are assumed to be rigid and static. The range finder is assumed to be capable of providing a depth map of the object in the scene. The range finder may have up to six degrees of freedom relative to the objects (three translations and three rotations). The displacements of the range finder can be described by rigid-body transformation. No prior knowledge of these transformations is assumed.

Our goal, then, is to design a system which can generate a complete 3-dimensional *b-rep* of a polyhedral object from a set of range images taken from different viewpoints [1], Figure 1.2. We assume that there exist overlapping surfaces between different views. The input to the system is a set of face models of visible surfaces extracted from the range images. A face model consists of the segmented faces and bounding edges traversed in a given direction (say the counter-clockwise direction). Face normals point outward. The methods for extracting a face model from a range image have been discussed intensively in the literature [27], [59]. In particular, Gupta and Zhu [22] deal with this problem in great detail. In this work, we will assume that the face model of visible surfaces of objects in a local view is given. The output of the system is a complete 3-dimensional *b-rep* of the object, as shown in Table 1.1 and Figure 1.2.E.

[1]Since no real range data are available, all the range images used in this project are synthetic.







A	B
C	D
E	

The input: multiple range images (A, B, C, and D) taken from different viewpoints. The output: a complete *b-rep* representation (E). The numbers in E are vertex labels.

Figure 1.2: Input and output of the proposed system.

Table 1.1: The output of the proposed system: a complete 3-dimensional *b-rep.*

vertex_T (*XYZ* Coordinate of vertices)

Vertex Label	X	Y	Z
1	74.00	10.00	50.00
2	80.00	40.00	80.00
3	74.00	10.00	80.00
4	80.00	110.00	80.00
5	20.00	110.00	50.00
6	20.00	110.00	80.00
7	26.00	80.00	100.00
8	74.00	80.00	100.00
9	26.00	48.00	100.00
10	74.00	48.00	100.00
11	20.00	40.00	80.00
12	26.00	10.00	80.00
13	20.00	40.00	50.00
14	26.00	10.00	50.00
15	80.00	40.00	50.00
16	80.00	110.00	50.00

f_v (vertices in a face)

Face Label	Vertex 1	Vertex 2	Vertex 3	Vertex 4	Vertex 5	Vertex 6
1	5	13	11	6		
2	11	13	14	12		
3	1	3	12	14		
4	3	2	11	12		
5	4	6	7	8		
6	6	11	9	7		
7	2	10	9	11		
8	8	7	9	10		
9	16	5	6	4		
10	1	14	13	5	16	15
11	2	4	8	10		
12	1	15	2	3		
13	15	16	4	2		

e_f (The two faces attached to an edge)

Edge Label	Start Point	End Point	L-Face	R-Face
1	2	3	4	12
2	3	1	3	12
3	5	6	1	9
4	6	4	5	9
5	6	7	6	5
6	7	8	8	5
7	8	4	11	5
8	8	10	8	11
9	10	2	7	11
10	7	9	6	8
11	6	11	1	6
12	9	11	6	7
13	9	10	7	8
14	2	11	7	4
15	11	12	2	4
16	12	3	3	4
17	12	14	2	3
18	1	14	3	10
29	13	11	2	1
20	14	13	2	10
21	5	13	10	1
22	1	15	10	12
23	15	16	10	13
24	16	4	9	13
25	16	5	10	9
26	4	2	11	13
27	15	2	13	12

1.3 Outline of Our Approach

Our system, starting from basic face models of visible surfaces of objects in each local view, extracts features, matches these features, generates rigid-body transformations that relate the local views, and finally merges these local views into a 3-dimensional *b-rep* description of the object. The diagram for the whole system is shown in Figure 1.3.

Since there exists a large number of matching candidates to be verified, an effective and efficient feature matching algorithm is designed to reduce the search time for finding the correspondences between views. In our matching algorithm, a *triple branch structure* is defined as a main feature to efficiently search the correspondence between views. This uniquely defined matching feature not only helps to reduce the cost of the matching process but also suffices to generate rigid-body transformation between views. A hierarchical searching strategy is realized by setting different constraints at different search levels. A prioritizing procedure is designed to arrange the order (priority queue) among matching candidates.

Based on these correspondences, the rigid-body transformations between views are generated and each local view is transformed into a global coordinate system, where the initial description is updated incrementally by integration of geometric information from each of the local views. A convenient and effective termination criterion is designed to monitor the integration process. This termination criterion also makes it possible to later integrate the whole system with a robust active vision mechanism. Finally, the complete *b-rep* description of the whole object is generated and can be easily converted into a winged-edge representation. Such a 3-dimensional

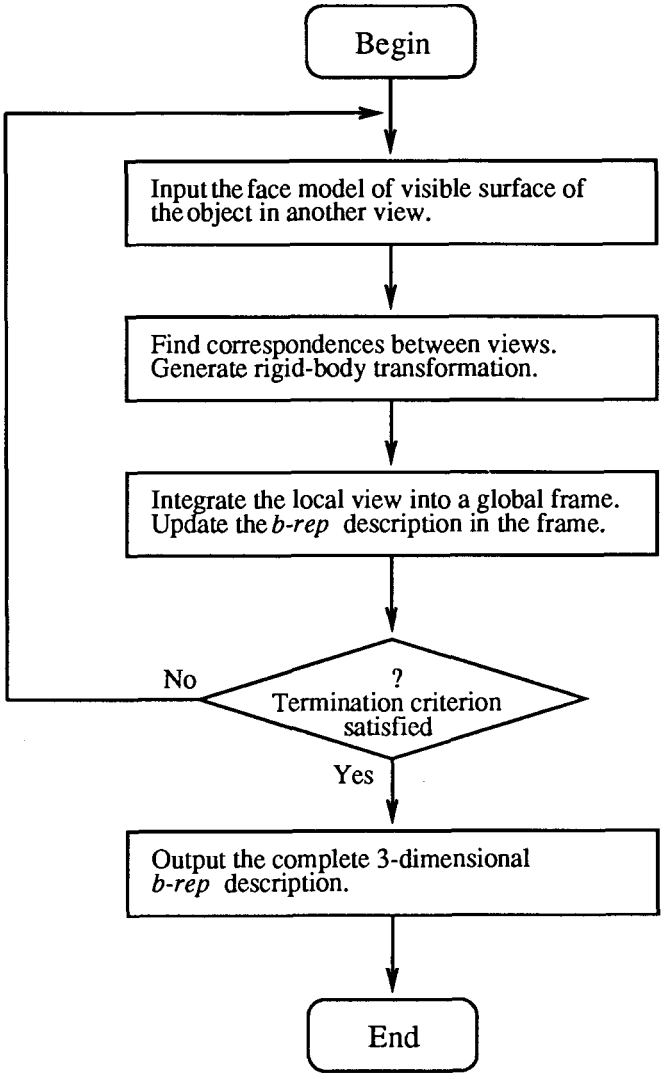


Figure 1.3: The diagram of the whole system.

description can then be used for the collision-free motion planning system presented in [21].

We have tested our system with synthetic range images. In the ideal case, with no noise present, the system indeed gives us a complete *b-rep* model, as expected. However, noise will always be present in any real system. We also tested our system with noise added to the synthetic range images. In our examples, with noise levels up to 5% of the maximum range values, the system was able to give correct output.

The main contribution of our work is summarized below. We propose a fast matching algorithm for matching views so that no prior knowledge of the transformations between views is required. In this matching algorithm, the *triple branch structure*, as a novel feature, was defined and used as the main matching feature. A convenient and effective termination criterion is set to automatically control the whole merging process. By integrating the processes of matching, transforming, and merging into a whole system, we have extracted the complete 3-dimensional *b-rep* description of a polyhedral object in the scene. A synthetic range image generator has also been implemented. Our system in conjunction with the synthetic range image generator also has applications in the area of robotics, *CAD* and geometric modeling.

1.4 Background and Literature Review

The process of 3-dimensional object description and recognition utilizes many techniques of image processing and computer vision, such as image acquisition, local feature extraction [28], correspondence finding, object pose determination [53] [54],

geometric transformation, geometric modeling, etc. The whole field of machine vision is concerned with these questions. In this section, we review various approaches, mainly in the following subareas: i) range image acquisition, ii) solid modeling, iii) matching, and iv) symbolic description from multiple views. In particular, we concentrate on the last two subareas. For general descriptions and reviews, please refer to [16], [23], [11], [47], [57], and [36].

1.4.1 Range Image Acquisition

Range-image acquisition is conceptually a simpler process than intensity-image formation [7]. At each pixel in a range image the depth value encodes information about (1) surface geometry and viewing geometry in terms of the distance from the sensor to object surface, and (2) the range finder characteristics (which include spatial resolution, range resolution, dynamic range, noise parameters, etc.). An important difference between an intensity image and a range image is that scene illumination and surface reflectance are not directly encoded in range image.

The range image or 3-dimensional information acquisition methods can be basically classified into two categories: active methods and passive methods. Active methods project energy onto a scene to measure range, whereas passive methods do not.

The most popular passive range finders utilize the stereo techniques [3], [17]. But one of the most difficult problems of this method is exactly matching the right-image and the left-image. Range from focusing [43] is another passive method which detects distance by measuring the degree of sharpness of the image. Many "Shape-

from-X" methods such as shape from motion [35], shape from shading [33], etc. can also supply the 3-dimensional information of objects to some extent. However, these methods may not extract dense and accurate depth maps.

Among active range image acquisition methods, laser energy is mostly used. Although some devices use ultrasound and radio wave techniques to determine range, their resolution is currently not as high as the laser range finder [37].

One of the common types of range finders is triangulation-based [46]. After a spot or line of light is projected onto an object, a camera or an infrared sensor is used to detect the light. Signal and image processing techniques are then used to determine the position of the spot or pieces of the line, and trigonometry is finally used to estimate distance.

Structured light range finders [37] are also among the most popular 3-dimensional sensing techniques which utilize various forms of structured light such as a ray, a sheet, a grid, or even cylinders, etc. The main advantages are simplicity and low cost.

1.4.2 Solid Modeling

The object representations commonly used by computer-aided-design (*CAD*) and geometric solid-object-modeling systems can be categorized [8], [7] as follows.

Wireframe representation: A wire frame representation of a 3-dimensional object consists of a 3-dimensional vertices list and edges list of vertex pairs. However, the wire frame representation is an ambiguous representation for determining the 3-

dimensional geometrical shape of an object.

Constructive Solid Geometry representation (*CSG-rep*): The *CSG* representation is specified in terms of a set of 3-dimensional volumetric primitives (blocks, cylinders, cones, and spheres are typical examples) and a set of Boolean operators, such as union, intersection and difference.

Spatial-occupancy representation: These representations use nonoverlapping subregions of the 3-dimensional space occupied by an object to define that object. The Voxel Representation is one of this type of methods, while the Octree Representation is another. Although Constructive Solid Geometry representation and Spatial-occupancy representation can unambiguously define an object's volume, the derivation of surface information from them is very computationally intensive.

Boundary representation (*b-rep*): This type of representation defines a solid object by defining the 3-dimensional surfaces that bound the object. All boundary representations contain a list of object surfaces and topological information which defines the relationships between surface patches. The winged-edge representation [5], [6], [14] is one very popular *b-rep* for polyhedra-liked objects. Each edge in the winged-edge data structure is represented by pointers to its two vertices, to the two faces sharing the edge, and to four of the additional edges emanating from its vertices. Each vertex has a backward pointer to one of the edges emanating from it, whereas each face points to one of its edges. The queries corresponding to nine types of *adjacency relationships*, such as which faces, edges or vertices are adjacent to each face, edge, or vertex, can be efficiently answered in the winged-edge representation. This representation makes it possible to determine in constant time which vertices or faces are associated with an edge. Another attractive property of the winged-edge

representation is that the data structures for the edges, faces and vertices are each of a small and constant size.

Since range images supply information about the surface of an object, it is more straightforward to generate a *b-rep* from those range images. Also, for path planning and the other robotic applications, we are more interested in the boundary surfaces of the object, and *b-rep* can explicitly supply the information about those.

1.4.3 Matching Methods

The problem of finding the correspondence in two images has been studied for a long time and still absorbs a great deal of interest in the computer vision community. The classical method is template matching by measuring correlation between 2-dimensional images [47]. Barnea and Silverman [4] proposed a method called SSDA.

Instead of directly matching intensity images, features are extracted first, and then correspondences are found between these features. Feature matching can reduce the search space and influence of noises. Among others, relaxation methods have been developed to solve a number of matching tasks [48], [51], [32]. Hough transform techniques have also been used for recognizing linear features and circular features in 2-dimensional case [49], [34], [2]. However, if Hough techniques are directly applied to vote for the six parameters of 3-dimensional rigid-body transformation, the accumulator size could become intractable or the precision could be greatly reduced. All these methods were first introduced to match 2-dimensional intensity images.

There are a few papers [42], [52], [12], [15], [19] discussing matching between

3-dimensional descriptions. However, most of them are concerned with the area of object recognition, in which the models of objects are known so that they can be used to guide and verify the matching process.

Oshima [42] proposes a system to recognize stacked objects using range data. The system describes a scene in terms of planes and smoothly curved surfaces. Models of objects are built in the system by showing them one at a time. Objects in an unknown scene are recognized by matching the description of the scene to those of the models. The matching program picks out regions which are most reliable and useful for recognition, and matches them to the regions of the models. Once candidate models are selected, the rest of the scene regions are searched for by guidance of the models.

In [52], a transformation is hypothesized by initially matching a few scene features with model features. The transformation is then tested with the rest of the features for verification. The matching process is based on a depth-first search of possible corresponding pairs of boundary components. In order to reduce the search space for speeding up the matching process, some methods are proposed. For example, the parts with simple shape should be ranked low because they may match many parts of other objects in a scene.

Faugeras [12] presents an efficient algorithm for 3-dimensional scene analysis. This algorithm uses a segmentation of the surfaces to be identified into geometrical primitives, the original data being obtained by a laser range finder. The algorithm estimates precisely the location and orientation of an identified object of the scene. In their matching algorithm, the primitives are planes and the consistency can be defined by $c(M, i_0, j_0) = |v_i \cdot v_{i_0} - v_j \cdot v_{j_0}|$, where (i, j) is the pair to be tested, (i_0, j_0)

is some pair in the previous matching, and the vectors v are the normals to the corresponding planes.

This relation comes from the fact that there exists a rotation consistent with the pairing $\{(i,j),(i_0,j_0)\}$, if and only if $v_i \cdot v_{i_0} = v_j \cdot v_{j_0}$. In other words, the constraint used here is that the angles between two surface normals in one view should be equal to the angles between the two corresponding surface normals in the other view. In their algorithm, two pairs of non-parallel planes are needed for estimating the rotation and three for the translation.

In references [19],[15], and [18], Grimson and his colleagues used local measurements of positions and surface normals to identify and locate objects in a scene. The objects are modeled as polyhedra having up to six degrees of positional freedom relative to the sensors. The approach operates by examining all hypotheses about pairings between sensed data and object surfaces and efficiently discarding inconsistent ones by using local constraints on distances between faces and angles between face normals. Their goal is to determine the power of simple geometric constraints in reducing the amount of search required to perform this task. While many other types of information can be used in recognition, Grimson et al focus exclusively on the geometric information available from a model.

They verified the possible matching between sensed data patch and model object face by checking the similarity of the angles between two face normals in the sensed data and in the model. They structured the search for consistent matches as the generation and exploration of an interpretation tree (IT) [25], [24], Starting at a root node, they constructed a tree in a depth first fashion. At the first level of the tree, they consider assigning the first measured patch to all possible faces; at the

next level, they would assign the second measured patch to all possible faces, etc. The number of possible interpretations in this tree, given s sensed patches and n surfaces, is as high as n^s. Therefore, it is not feasible to explore the entire search space in order to apply a model test to all interpretations.

Grimson et al's algorithm exploits local geometric constraints to remove entire subtrees from consideration. They use *heuristic Search Ordering* to find the best possible interpretations or paths in the *IT*. The combined area of data patches of an interpretation is chosen as the quality of measurement. The interpretation with the largest combined area is considered to be the best interpretation. In addition to the heuristic searching to prune the interpretation tree, Hough transform is used as a coarse filter to reduce the size of the initial interpretation tree.

During the *Model Test*, the feasible interpretations are tested for consistency with surface equations obtained from the object models. An interpretation is legal if it is possible to solve for a rotation and translation that would place each sensed patch on a model surface. The sensed patch must lie inside the model face, not just on the surface defined by the model face equation.

It is clear that the object model is used both in generating feasible interpretations and model testing. Our system does not presume that the object model is given and furthermore we will build this model. In the matching algorithm, we select different features and local geometrical constraints for matching views.

1.4.4 Symbolic Description from Multiple Views

Hong and Shneier [31] propose a method which involves using intensity images acquired from arbitrary but known locations to construct the spatial representation incrementally. This is accomplished by projecting the image resulting from each view into the 3-dimensional world, and intersecting the views in the following way.

Each object in the image projects into the world as a "cone" with its tip at the center of focus of the lens, and its cross section defined by the boundary of the object. When two images are acquired from different viewpoints, an object appearing in both images is constrained to lie in the intersection of the cones from the viewpoints. If it has been seen many times from different viewpoints, then not only will its position be more tightly constrained, but so will its shape. Eventually, the whole workspace should be represented in a way that closely approximates its true state. However, Hong and Shneier do not explicitly supply a symbolic description. The related works can also be found in [41],[6].

Some researchers have implemented systems that construct 3-dimensional objects from multiple stereo views [29], [56], [30],[44]. Herman, Walker and Kanade present a system (MOSAIC) that incrementally reconstructs a complex 3-dimensional scene from a sequence of intensity images obtained from multiple viewpoints. The system encompasses several levels of vision process, starting with images and ending with symbolic scene description. The system includes stereo analysis, monocular analysis, and constructing and updating the scene model.

Each view of the scene, which may be either a single image or a stereo pair, undergoes analysis which results in a 3-dimensional wire-frame description that rep-

resents portions of edges and vertices of objects. The model is a surface-based description constructed from the wire frames. With each successive view, the model is incrementally updated and gradually becomes more accurate and complete. In their paper, the authors extensively discuss the process of geometric reasoning and how to make hypotheses on the parts of objects which cannot be seen from the images available so far. However, instead of implementing the registration mechanism which can supply the information to generate the rigid-body transformation from new local view to the global view or between local views, they simply assume this type of transformations is known. Hence, to further complete the system, a robust general matcher needs to be developed. Since the input data is from either a single intensity image or an image pair, the process of extracting the local wire-frame data is relatively complicated, and matching in the stereo image is very crucial.

Some researchers have used range images directly as the input. Henderson [26] has developed a method for finding planar faces in range data. In his method, a list of 3-dimensional object points is assumed to be given by a range finder. To handle multiple depth maps, points are transformed into one project-centered coordinate system using transformation data recorded during range-image formation. These points are stored randomly in a list with no topological connectivity information. The points are then organized into a 3-dimensional binary tree. Next, each point's neighbors are determined with the aid of the 3-dimensional tree, and the results are stored in a 3-dimensional spatial approximation graph. Then, a spiraling sequential planar-region-growing algorithm is used to create convex planar faces using the spatial proximity graph as input. The union of these faces forms the polyhedral object representation as extracted from the range data. In this method, the transformations between views are assumed known.

Stenstrom [50] describes a method for constructing a full and reasonably accurate wire-frame description of an object by directly employing range information from multiple views. The objects are placed on a turntable whose center is fixed in the camera's field of view and a set of range images is acquired by the rotation of the turntable. By using the known rotating angles, the transformation between views can be generated. Then, line features are extracted in each view and finally merged in one common frame.

Vemuri and Aggarwal [55] utilize both the range image and intensity image to construct 3-dimensional model from multiple views. Their technique for integrating the information from multiple views does not require the correspondence relationship between views to be determined. The object for which the model is to be constructed is assumed to rest on a plane (base plane). A pattern consisting of a single line is drawn on the base plane. By observing the orientation of the base plane pattern in the intensity images from multiple views, the interframe transformation required to register any two views in a common reference coordinate system is derived. More papers about utilizing various sensory devices can be found in the review of using multisensory images to derive the structure of 3-dimensional objects [40].

Xie [58] proposed an interactive expert system for reconstructing 3-dimensional obstacles based on rule based inference. The purpose of these rules was to eliminate highly improbable interpretations.

Kim and Connolly [39],[38], and [9] present a method for utilizing the multiple views efficiently. Instead of selecting viewpoints randomly, they actively determined the next best viewpoint by setting up a set of optimization criteria so that the system can get the maximum amount of information from the least number of viewpoints.

Active vision may provide a robust mechanism for reconstruction and recognition of 3-dimensional objects.

Among multiple view model construction methods, quite a few assume that the transformations between views are known a priori or can be generated with the help of prespecified conditions. To make these methods more robust, more sophisticated matching methods should be used. In our approach, we design an efficient matching algorithm to find the correspondences between views, define an effective termination criterion to dynamically control the merging process, and integrate the processes of matching, transforming, and merging into a whole system.

1.4.5 Organization

The rest of this thesis is organized as follows.

In Chapter 2, a mechanism for hierarchical matching is presented. Triple branch structure features are first extracted. Then a hierarchical searching algorithm is applied to efficiently find the correspondences between views. The assumption for this matching algorithm is that there exist two common faces between views.

Chapter 3 deals with the issue of correspondence versus rigid-body transformation. The minimum number of correspondences to generate rigid-body transformation is discussed and practical algorithms are given.

Chapter 4 proposes an algorithm for integrating multiple local views and gradually updating the global description. A convenient and effective termination criterion is also given.

Chapter 5 presents some experimental results and analyzes the performance of the system.

Finally, Chapter 6 summarizes our work. Appendix A describes a synthetic range image generator which utilizes face normals to distinguish the visible faces from non-visible ones and uses the Z-buffer technique to process partially visible faces.

CHAPTER 2

Matching Views

In this chapter, we propose a method for determining the correspondence between two views, assuming that the position of each viewpoint and the camera setting parameters for each viewpoint are completely unknown. The input to the matching algorithm is face models of visible faces in local views. A point (vertex) in one view and a point (vertex) in another view construct a point pair. Our goal is to find the correctly matched point pairs (corresponding point pairs) in two different views. Our basic approach is to use geometric features that are invariant under rigid-body transformation.

A brute force search would compare each of the distances between the vertices in one view with each of the distances between the vertices in another view to find the correspondence. Assuming there are M edges (M distances) in one view, and N edges in another view, the complexity of brute force search between two views may be as high as N^M. If we treat each of these edges separately, the search time can be reduced, but then the probability of false matching will be increased

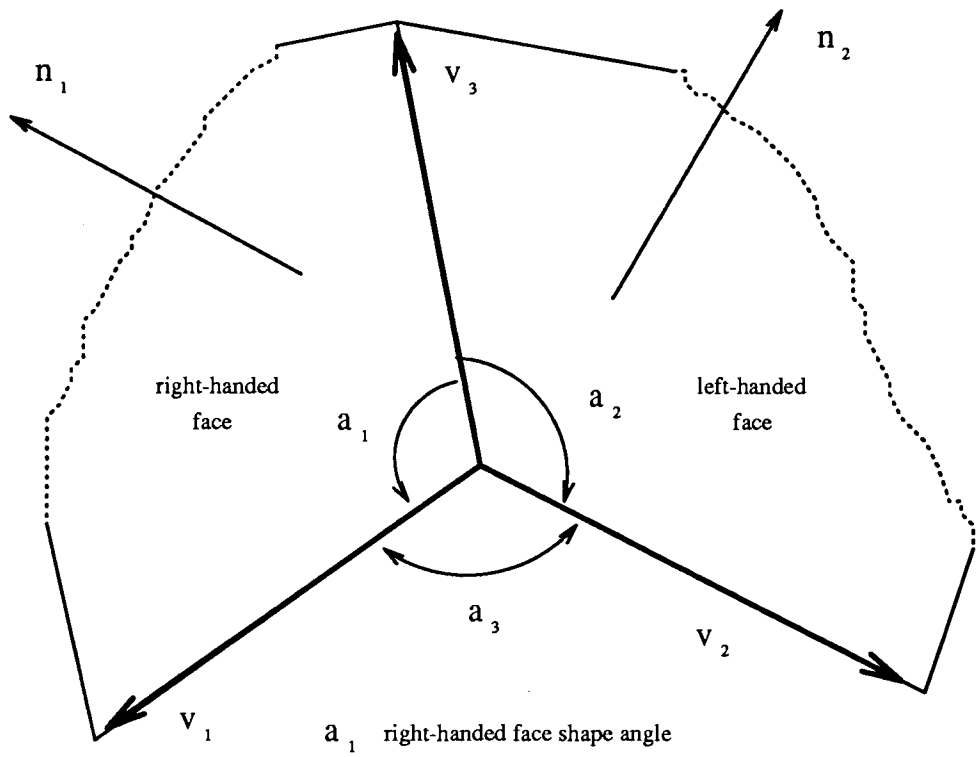
dramatically. Instead, we chose certain type of features with particular structures which can help the efficient matching algorithm and generate correct rigid-body transformation between views. The complexity of our matching algorithm is $O(N^2)$. In the feature matching process, the essential point is a hierarchical search strategy. By introducing different constraints at different search levels, we can narrow down the number of candidate features at each search level. The basic feature chosen to be matched between views in our algorithm is the TRIPLE BRANCH STRUCTURE. The geometric constraints used in the hierarchical sequence from top to bottom are: 1) the angle between two face normals, 2) the face shape angles in the triple branch structure, and 3) the distance between vertices.

2.1 Triple Branch Structure

Definition

A *triple branch structure* as shown in Figure 2.1 consists of three intersecting and noncoplanar edges. Each of the edges is called a *branch.* In our case, the triple branch structure is constituted by two noncoplanar faces (the faces are not necessarily fully visible). The branch (intersection) constructed by the two faces is called the *principal branch.* The rest of the branches are called the *subordinate branches.* The intersection point of the three branches is called the *triple corner.* All three branches are denoted as vectors whose origins are at the triple corner. The magnitudes of the vectors are equal to the lengths of the corresponding edges respectively. The principal branch is denoted as the *principal vector* $\vec{v}_3$.

The two faces, which constitute the triple branch structure, are denoted by the



a_1 right-handed face shape angle

a_2 left-handed face shape angle

a_3 principal angle

v_1 right-handed face vector

v_2 left-handed face vector

v_3 principal vector

n_1 right-handed face normal

n_2 left-handed face normal

Figure 2.1: Triple branch structure.

right-handed face of the triple branch structure and the left-handed face of the triple branch structure. The face whose normal (pointing outward) and surrounding edge loop (following the direction pointed by the principal vector) meet the right-handed (left-handed) system criterion is called the *right-handed (left-handed) face of the triple branch structure.*

The branch which is in the right-handed face is defined as *right-handed vector* $\vec{v}_1$. Its origin is at the corner of the triple branch structure.

The branch which is in the left-handed face is defined as *left-handed vector* $\vec{v}_2$. Its origin is at the corner of the triple branch structure.

The angle between the right-handed face normal and the left-handed face normal is called the *principal angle.*

The angle between the principal vector and the right-handed (left-handed) vector is called the *right-handed (left-handed) face shape angle.* Note that this angle is inside angle of the face.

Two triple branch structures are called *similar* when the corresponding principal angles, the right-handed face shape angles and the left-handed face shape angles, are the same.

Two similar triple branch structures are called *identical* when the magnitudes of the corresponding vectors are the same.

A *triple pair* is formed by a triple branch structure in one view and a triple branch structure in another view. If the two triple branch structure in a triple pair are identical, this triple pair is called the corresponding (or matched) triple pair. A

matched triple pair has four matched point pairs.

There are several reasons for using the triple branch structure as our main geometric feature. First, the triple branch structure is a very natural feature in the polyhedral scene. As long as two adjacent faces (even two adjacent partial faces) exist, so does the triple branch structure. Note that a triple branch structure does not require that angles among the three branches be orthogonal angles. Its application goes beyond the trihedral world. This feature can also be generalized for curved objects.

Second, two matched triple branch structures can sufficiently supply four non-coplanar corresponding point pairs to generate a rigid-body transformation (see Chapter 3 for details).

Third, the definition of the right-handed (left-handed) face of a triple branch structure is independent of viewpoints. It is a fact that if in one view, a face of a triple branch structure is defined as a right-handed (left-handed) face, then in other views, the corresponding face of the corresponding triple branch structure is guaranteed to be assigned as a right-handed (left-handed) face without any information about correspondence between the views. Since the right-handed (left-handed) vector is associated with the right-handed (left-handed) face, it possesses the same property. This invariance property is very helpful, e.g. it can reduce the number of triple pairs by a factor of two compared with the case where the two subordinate branches are arbitrarily labeled among views.

Fourth, we could define more complex features as the basic matching features, but the more complex the feature is, the more search time is needed. Because we are

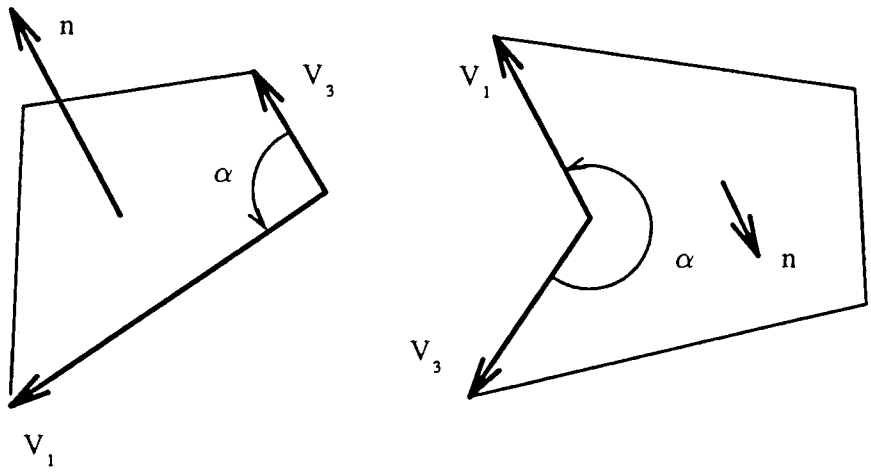
using a hierarchical search strategy, we do not need to find an exact match at the low search levels. Instead, our purpose is to narrow down the number of candidates for the next search level. Hence, we chose the triple branch structure as the basic matching feature.

Fifth, we could use three non-intersecting branches. For example, instead of only checking the angles between two adjacent faces, we could check the angles between any two faces [19]. However, this approach could bring in some weak points. First, it obviously increases the number of basic matching features so that it costs more search time. Secondly, when two faces are far away from each other, it is very possible that one of the corresponding faces will be out of each other's view, so that a search for such types of face pairs becomes unnecessary.

Finally, we could use one face instead of two adjacent faces as a basic matching feature. But this also increases the search time and brings in more uncertainty or false matching. However, in case the triple branch structure does not exist, this can be one of the choices and the hierarchical matching strategy can still be used.

When a matching algorithm uses the triple branch structure as the basic matching feature, the following assumption must be satisfied so that the possible matching can be found:

> In any two views, there exists at least one pair of adjacent faces (or partial faces) in common, i.e. these two faces in one view correspond to two faces in another view.



α right-handed face shape angle

V_1 right-handed face vector

V_3 principal vector

n right-handed face normal

Figure 2.2: Face shape angle.

A note about distance and angle calculation is in order here. The Euclidean distance is directly used to measure distances. To reduce ambiguity in derived angles, we define a face shape angle α inside a face, see Figure 2.2. Since subroutine *atan2(y,x)* has a 180^0 ambiguity, we propose the following method, which utilizes the existing information about face normals.

Let face shape angle be α, principal vector $\vec{\omega}$, left-handed(or right-handed)face vector $\vec{v}$, and face normal $\vec{n}$

step 1 Calculate angle $\tilde{\alpha}$ between vector $\vec{\omega}$ and vector $\vec{v}$ and take the angle in the first quadrant.

$$\tilde{\alpha} = cos^{-1}\frac{\vec{\omega}\vec{v}}{\|\vec{\omega}\|\|\vec{v}\|} \quad 0 \leq \tilde{\alpha} \leq \frac{\pi}{2}$$

step 2 Decide in which quadrant $\vec{v}$ is in respect to $\vec{\omega}$

Let $\vec{\omega} = \vec{X}$ and $\vec{n} = \vec{Z}$. First, determine $\vec{Y}$

$$\vec{Y} = \vec{Z} \times \vec{X}$$

Then, decide which quadrant of this XYZ frame $\vec{v}$ is in.

$$sign_1 = sign(\ \vec{v}\vec{X}) \quad and \quad sign_2 = sign(\ \vec{v}\vec{Y}) \text{ where}$$

the function *sign* takes the sign of its argument.

step 3 Determine the face shape angle

$$\alpha = \begin{cases} \tilde{\alpha} & \text{if } sign_1 \geq 0 \text{ and } sign_2 \geq 0 \\ \pi - \tilde{\alpha} & \text{if } sign_1 \leq 0 \text{ and } sign_2 \geq 0 \\ \pi + \tilde{\alpha} & \text{if } sign_1 \leq 0 \text{ and } sign_2 \leq 0 \\ 2\pi - \tilde{\alpha} & \text{if } sign_1 \geq 0 \text{ and } sign_2 \leq 0 \end{cases}$$

2.2 Hierarchical Search Algorithm

The basic idea of our hierarchical search algorithm is that the search for the corresponding triple branch structures is implemented at different levels. At each finer search level, new and more stringent constraints are added so that the number of matching candidate triple branch structures becomes fewer. At the same time, the match probability of each of the corresponding triple branch structures gets progressively higher.

Since the calculation of angles between edges does not need the complete edge information, only partial edges will do. However, the calculation of the length of an edge needs both end points of this edge. The distance constraint is more stringent than the angle constraint. Because of that, we check the angle match before the

distance match so that even in the case of partially visible faces, the search for the match can still be carried out to a certain level and supply a relatively small number of possible matching candidates.

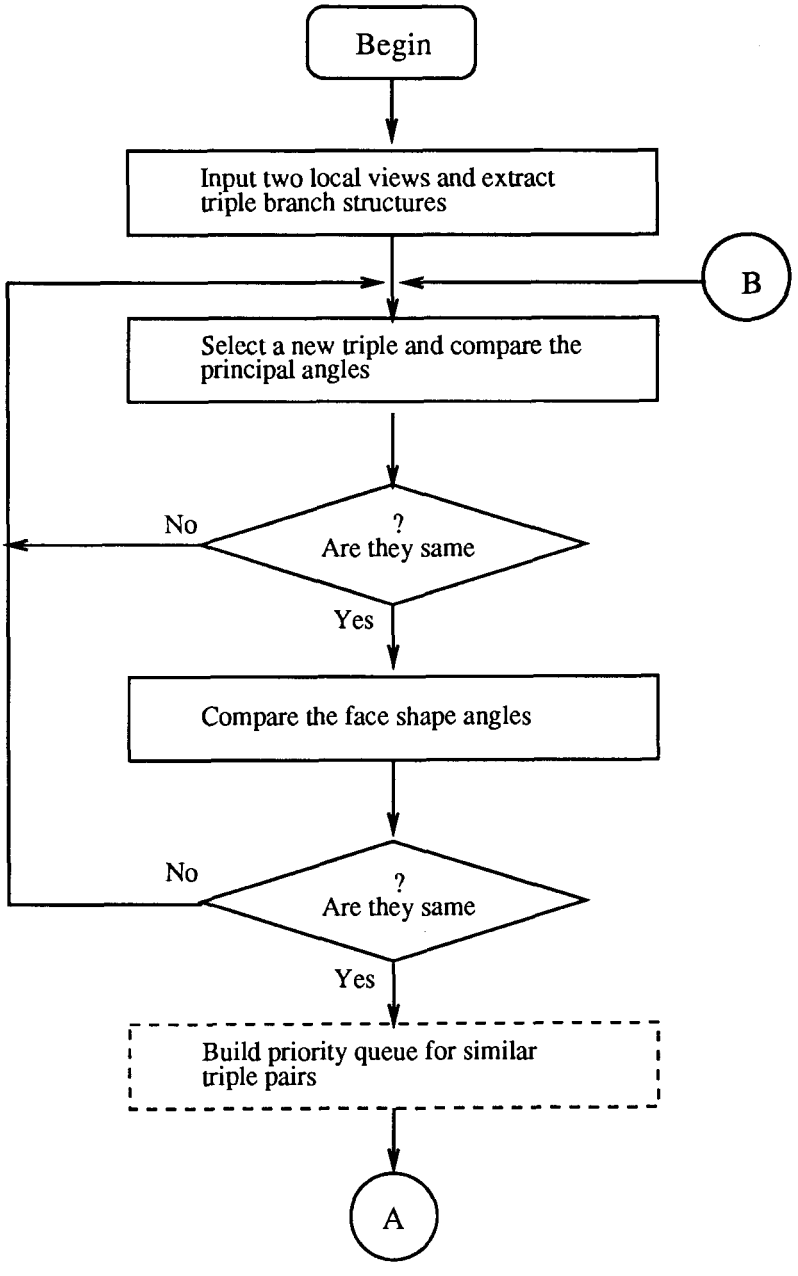
Our matching algorithm also features "priority queue" and "feedback verification". In the prioritizing procedure, the potentially matched point (vertex) pairs between views become more evident. The feedback verification procedure is used to confirm that the finally resulting potential correspondences are the best ones among all. Figure 2.3 shows the flowchart of this algorithm. Let us now discuss it in some detail.

2.2.1 Similarity Check

Matching principal angles

At this search level, the principal angles of the triple branch structure in two views are selected as a constraint. For each of the principal angles $\alpha_{3,I}$ in one view, the algorithm tries to find all possible matching principal angles, say $\alpha_{3,J}$ in another view. If the triple branch structures are the same, the principal angles of them must be the same. To handle the case where some distortions of the original data exist, (it may occur when any type of noises or calculation errors are introduced in any of the processing stages), we define two angles being the same within a small variation $\pm\delta$, where $\delta \geq 0$.

Note that the condition that the same triple branch structure must have the same principal angles is only a necessary condition. Although there may exist some false matches in the resulting candidates, the number of candidates to be checked



The dotted-line frame indicates where the priority queue can be applied.

Figure 2.3: Hierarchical search algorithm — continued.

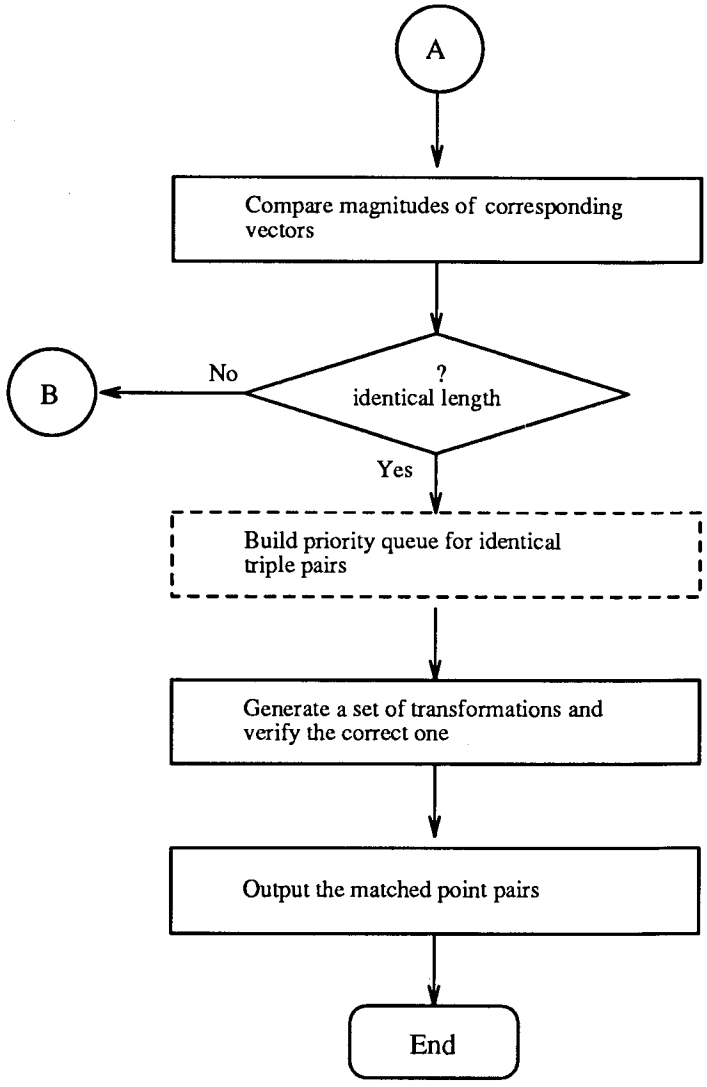


Figure 2.4: Hierarchical search algorithm.

at the next level is reduced.

Matching face shape angles

Since the right-handed face shape angle and the left-handed face shape angle of a triple branch structure are uniquely defined, and are independent of rigid-body transformation, we need only to check the similarity between the right-handed (left-handed) face shape angles in one view and the right-handed (left-handed) face shape angles in another view. At this level, the face shape angles are added as the new constraints into the similarity measurement.

As defined in the above subsection, the face shape angles reflect the local shape of the faces which the triple branch structure lies on. The same faces must have the same local shapes. A small variation ($\pm\delta, where\delta \geq 0$) is also introduced in the process of checking similarity faces shape angles to increase the algorithm's tolerance of noise and calculation error.

The condition that the same triple branch structures must have the same right-handed face shape angles and the same left-handed face shape angles is a necessary condition and not a sufficient condition. It only means that if any of the two corresponding face shape angles are not the same, the triple branch structure can not be the same. But, by using the face shape angle constraint, we can further reduce the number of matching candidates for the next search level without losing the correct matching candidates.

2.2.2 Identity Check

The invariance of distances between vertices is another intrinsic feature of rigid-body transformation. At this search level, the magnitudes of vectors are used as the new and more stringent constraint to verify the identical triple branch structures. If the corresponding magnitudes of two similar triple branch structures are the same, then these two triple branch structures are identical and they are the potentially matched candidate or one of the matched candidates if there is more than one identical triple branch structure in one view.

2.2.3 Priority Queue

Since the identical triple pairs are still the candidates for matching views, we need to verify them. Instead of randomly selecting the candidate triple pairs for verification, we build a priority queue to find the most likely corresponding point pairs and use these pairs as the candidates to determine the correct match between views. We may reduce the computational time for the verifying procedure (which is usually time-consuming) by pruning this queue rather than exhausting all the potentially matched triple branch structures.

The earlier set of geometric constraints considered only matches between individual triple pairs. The constraints can be strengthened by propagating the effect of a legal match to match the point pairs in its neighboring triple pairs. More weight should be put on the point pairs that have been shown matched pairs by several potentially matched triple pairs. It is likely that one point pair is mismatched because of one mismatched triple pair. However, it is much less likely that a mismatched

point pair will appear simultaneously in several independently matched triple pairs (it is less likely that all the mismatched triple pairs made the same mistake). In general, only can the correctly matched point pair be shown in several matched triple pairs. By using the following prioritizing procedure, we virtually group the locally linked triple branch structures (two triple branch structures share at least one branch) and find the potentially matched point pairs whose matches are given by the group of matched triple pairs. Here is the prioritizing procedure:

1. Each candidate triple pair votes its four corresponding point (vertex) pairs (the end points and starting point of the three vectors).

2. An accumulate array is used to record each vote. For each voting, the score of the corresponding point pair is incremented by one. For example, if there are N triple pairs containing the same point pair, then the score for this point pair is N. In other words, there exist N potentially matched triple pairs which indicate that the two points (vertices) match each other.

3. After the score is accumulated for each point pair, the vote for each of the candidate triple pairs is determined by adding the scores of its four point pairs.

4. The triple pair with the highest score is selected as the first one to be verified in the later verifying process, the triple pair with the second highest score is chosen as the second one to be verified, then the third, and so on, until certain termination criteria are met.

The figures in the Chapter 5 show the voting results in experiments. The voting procedure can be applied to the verification of either the similar triple branch

structure pairs or identical triple branch structure pairs.

2.2.4 Feedback Verification

Through the previous processes, the number of the identical triple branch structures is small. However, if there is more than one triple branch structure identical to the one in another view, then a wrong transformation could be deduced based on the false correspondences. To avoid this, we use the following verifying process:

1. Use one of the candidate corresponding triple branch structures to generate a transformation matrix.

2. Transfer [1] the points in one view (view A) into the other view (view B) by using this transformation matrix ${}^{B}T_{A}$. Then, calculate the number of points which are from view A and overlap with the points in view B, by the distance $||{}^{B}\vec{Q}_i - {}^{B}T_{A}\,{}^{A}\vec{P}_i||$, where ${}^{B}\vec{Q}_i$ is the point in view B and ${}^{A}\vec{P}_i$ is the point in view A. Once one of such distances is smaller than a threshold, the score for this transformation is increased by one.

3. Repeat steps 1 and 2, for each of the candidate corresponding triple branch structures.

4. Select the transformation with the highest score as the most likely correct transformation between the views, if its score is larger than the predefined

[1] Throughout this thesis, by the phrase "transforming a point from frame A to frame B" we mean transforming the coordinate of the point observed from frame A to frame B.

threshold.

It is obvious that the method to verify the correct match at this stage is very time-consuming and sometimes so much time is required that it is not possible to use a brute force search method.

Fortunately, in this hierarchical matching algorithm the number of candidates to be verified at the last stage generally becomes so small that the feedback checking turns out to be affordable. Another merit of this algorithm is that even if for some reason, the matching measurement stops at a certain intermediate search level, we can still get a relatively good correspondence. For example, if we do not have enough information to calculate the length of edges so that we cannot really verify the identity of the triple branch structure, but the potentially matched similar triple branch structures still provide very good information. In many cases, based on this correspondence information, we can still derive the correct rigid-body transformations.

CHAPTER 3

Correspondence and Transformation

This chapter discusses the theory of generating rigid-body transformations from some known corresponding point pairs between two local frames and algorithms to calculate the transformations in practice. A variety of literature exists on this topic [12], [1]. Some of the results presented in this chapter may be known implicitly. However, we believe that it is important to state them explicitly. A corresponding point pair means a pair of 3-dimensional coordinates $({}^{A}\vec{P}, {}^{B}\vec{Q})$ [10] of the same point in the 3-dimensional space observed from different co-ordinate frames (frame A and frame B) related by a rigid-body transformation ${}^{B}T_{A}$, ${}^{B}\vec{Q} = {}^{B}T_{A}\,{}^{A}\vec{P}$. A homogeneous transformation matrix is used as a representation of rigid-body motion, since this representation is easily manipulated by matrix operations and is often used in the robotics world. Note that an arbitrary homogeneous transformation represents a larger set of transformations (including perspective transformation, scaling, etc.)

than just the set of rigid-body transformations.

Assume that there exists a set of corresponding point pairs $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ to\ N\}$, where $\{{}^{A}\vec{P}_i, i = 1\ to\ N\}$ and $\{{}^{B}\vec{Q}_i, i = 1\ to\ N\}$ are the coordinates of a set of rigid 3-dimensional points observed from different frames (frame A and frame B) related by a rigid-body transformation ${}^{B}T_A$. The problem then is as follows: based on a subset $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ to\ M\}$, where $M \leq N$, of the corresponding point pairs, generate the rigid-body transformation ${}^{B}T_A$ which can satisfy the whole set of corresponding point pairs.

The least-squares-fit (*LSF*) method is commonly used [13] to deal with a case where a large number of corresponding point pairs is available. However, in some cases, the number of available corresponding point pairs may be very small (depending on the shape of objects, objects' layout, viewing angle, etc). Hence, it becomes more important to know explicitly what is the minimum number of corresponding point pairs for generating such a rigid-body transformation that satisfies the whole set of corresponding point pairs.

3.1 Transformation and Four Corresponding Point Pairs

We are given a set of corresponding point pairs $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ to\ N\}$, as defined at the beginning of this chapter. Under what circumstances, is a transformation which satisfies a subset of the corresponding point pairs of this set, the rigid-body transformation ${}^{B}T_A$ for the whole set of the corresponding point pairs?

Lemma 1: [1] An arbitrary transformation is a rigid-body transformation ${}^{B}T_{A}$, which transforms all the points in frame A to their corresponding points in frame B, if and only if this transformation satisfies at least four non-coplanar corresponding point pairs among the set $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ \ to\ \ N\}$.

This lemma implies that based on four non-coplanar corresponding point pairs in the set $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ \ to\ \ N\}$ the rigid-body transformation ${}^{B}T_{A}$ can be derived as below.

Given four non-planar corresponding point pairs between frame A and frame B, $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ \ to\ \ 4\}$.

$$ {}^{B}T_{A} = M_Q M_P^{-1} $$

where

$$ M_P = \begin{pmatrix} \vec{P}_1 & \vec{P}_2 & \vec{P}_3 & \vec{P}_4 \end{pmatrix} \ \ and \ \ M_Q = \begin{pmatrix} \vec{Q}_1 & \vec{Q}_2 & \vec{Q}_3 & \vec{Q}_4 \end{pmatrix} $$

Since M_P is composed of four non-coplanar points, the rank of the matrix M_P is equal to 4. So the inverse matrix M_P^{-1} always exists and the solution ${}^{B}T_{A}$ is unique.

---

[1] The proof is given in Appendix B.

3.2 Minimum number of Corresponding Point Pairs

It is assumed that there exists a set of corresponding point pairs, $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ \ to\ \ N\}$, where ${}^{A}\vec{P}_i$ and ${}^{B}\vec{Q}_i$ are the coordinates of a set of rigid 3-dimensional points observed from different frames related by a rigid-body transformation ${}^{B}T_A$. The following lemma gives the minimum number of corresponding point pairs needed to generate the rigid-body transformation ${}^{B}T_A$.

Lemma 2: At least three non-colinear corresponding point pairs are needed to generate the rigid-body transformation ${}^{B}T_A$.

Note, however, that an arbitrary transformation which satisfies three non-colinear corresponding point pairs is not necessarily a rigid-body transformation for all the points between frames. The correctness of Lemma 2 is proved by the following discussion.

Since in the $3D$ space, rigid-body transformation can be specified by six parameters (three rotation parameters and three translation parameters), there are only six unknowns to be determined to solve the transformation equation.

One misleading argument could be that since two points in the $3D$ space have six coordinates, it follows that two corresponding point pairs are enough to generate the rigid-body transformation; using these six coordinates, one can write six equations for those six unknowns. However, it can be proved as follows that those six equations are not independent. Under the assumption that frame A and frame B are related

by a rigid-body transformation ${}^{B}T_{A}$, the distance ${}^{A}D_{ij}$ between any two of the points in frame A is the same as the distance ${}^{B}D_{ij}$ between the corresponding points in frame B, i.e., ${}^{A}D_{ij} = {}^{B}D_{ij}$. From this it follows, that

$$ {}^{A}z_{j} = {}^{A}z_{i} \pm \sqrt{{}^{B}D_{ij}^{2} - ({}^{A}x_{i} - {}^{A}x_{j})^{2} + ({}^{A}y_{i} - {}^{A}y_{j})^{2}}. $$

Similarly, we can derive the relationship between any one of the six coordinates and the remaining five. It is clear that among six coordinate pairs from two corresponding point pairs, only five coordinate pairs are independent and the other one is dependent on the rest. Therefore, two corresponding point pairs are not enough to generate the rigid-body transformation ${}^{B}T_{A}$.

Now, let us investigate the case when there are three corresponding point pairs $({}^{A}\vec{P}_{i}, {}^{B}\vec{Q}_{i})$, $({}^{A}\vec{P}_{j}, {}^{B}\vec{Q}_{j})$ and $({}^{A}\vec{P}_{k}, {}^{B}\vec{Q}_{k})$.

Among three non-colinear corresponding point pairs in $3D$ space, there are nine coordinates. However, there are three distance constraints based on the assumption of rigid-body transformation, i.e., ${}^{A}D_{ij} = {}^{B}D_{ij}$, ${}^{A}D_{jk} = {}^{B}D_{jk}$ and ${}^{A}D_{ik} = {}^{B}D_{ik}$. After the distance constraints are applied, there are only six independent coordinate pairs left. Since we only need to derive six unknown parameters for rigid-body transformation, we can write down six independent equations to solve the problem.

Note that the six equations, in terms of three rotation parameters and three translation parameters are non-linear equations. A general analytic solution for such non-colinear equations is not known. Hence, numerical techniques are used but they may not guarantee a solution.

We present a practical method which needs only three non-colinear corresponding point pairs and linear mathematical calculation to generate the rigid-body transfor-

mation. In this method, a hypothetical corresponding point pair is constructed as the fourth corresponding point pair. There are many methods to generate the fourth point if this point is not co-planar with respect to the three points given, and has a unique relationship with these three points. In our method, the cross product of two vectors determined by the three points given is used to generate the fourth point. Briefly, the method is:

Given three non-colinear corresponding point pairs between frame A and frame B, $\{({}^A\vec{P}_i, {}^B\vec{Q}_i),\ i = 1\ \ to\ \ 3\}$

1. Generate the fourth point pair $\{{}^A\vec{P}_4, {}^B\vec{Q}_4\}$

   $$\vec{P}_4 = \vec{P}_1 + {}^A\vec{W}_3 \qquad \text{and} \qquad \vec{Q}_4 = \vec{Q}_1 + {}^B\vec{W}_3$$

   where

   $${}^A\vec{W}_3 = {}^A\vec{W}_1 \times {}^A\vec{W}_2, \quad {}^A\vec{W}_1 = {}^A\vec{P}_2 - {}^A\vec{P}_1, {}^A\vec{W}_2 = {}^A\vec{P}_3 - {}^A\vec{P}_1$$

   $${}^B\vec{W}_3 = {}^B\vec{W}_1 \times {}^B\vec{W}_2, \quad {}^B\vec{W}_1 = {}^B\vec{Q}_2 - {}^B\vec{Q}_1 \ , {}^B\vec{W}_2 = {}^B\vec{Q}_3 - {}^B\vec{Q}_1$$

2. Derive the rigid-body transformation

   $${}^BT_A = M_Q M_P^{-1}$$

   where M_Q and M_P are defined as in section 3.1.

3.3 Best Estimate

Since the original data may be distorted by many kinds of noise (such as sensor noise and moving parameter measurement error), one transformation may not fit all the corresponding point pairs. In this case, we need an estimation process to reduce the affect of noise (assuming that noise is zero mean). A least-squares-fit method

is used to generate the estimation of transformation between views. In most cases, the greater the number of corresponding point pairs used, the more accurate is the derived transformation.

The estimated transformation [13] is given below

$$T = (\sum_{i=1}^{N} {}^{B}\vec{Q}_i {}^{A}\vec{P}_i^T)(\sum_{i=1}^{N} {}^{A}\vec{P}_i {}^{A}\vec{P}_i^T)^{-1}$$

where N is the number of corresponding point pairs. From Lemma 1, it follows that $N \geq 4$. ${}^{A}\vec{P}_i$ and ${}^{B}\vec{Q}_i$ are the corresponding points in local frame A and local frame B respectively. Note that when the influence of noise is reduced to zero, this method is still valid and generates the correct transformation between views.

In some controlled situations, parameters of rotation and translation are given (e.g., the motion parameters for the displacement of PUMA arm are known). However, even in such cases, errors often exist in the transformation matrix. These errors in the transformation matrix can be eliminated using the LSF. To do this, we first transfer the local views into a global frame. Then in the global frame, we find as many "identical vertices" as possible, (see Chapter 4). Then we refine the transformation using the LSF method.

CHAPTER 4

Merging Multiple Views

In this chapter, we present a method for merging multiple views. The main function of the merge module is not just transferring all the features in the different views into a common frame, but integrating selectively the features and local symbolic descriptions coming from different views. To build a complete description of a whole object, we assume that each face of the object appears in at least one local view.

The flow chart for merging is shown in Figure 4.1. First, all the features (vertices, edges, faces, etc.) are transferred from their local frames (viewpoint centered frames) to a universal global frame based on the estimated rigid-body transformations. Then, duplicated features are thrown away and new features are added to the symbolic description in the global frame. A termination criterion is used to control the completion of integration processing. In the following sections, we discuss 1) similarity measures between features, 2) the termination criterion, and 3) the integration algorithm.

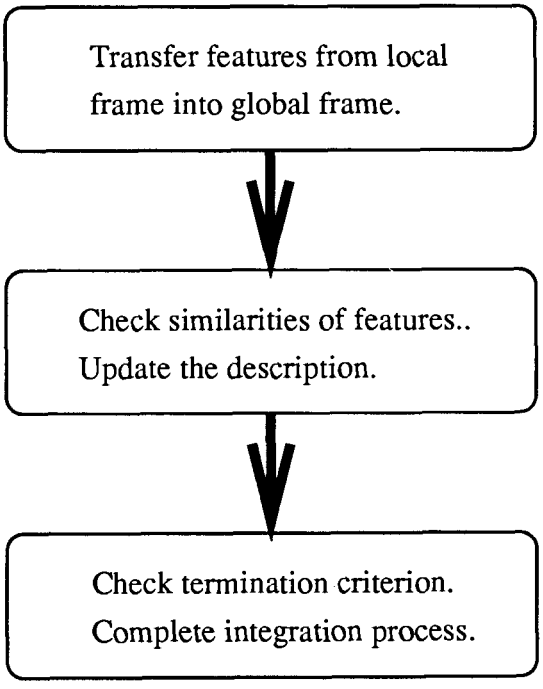


Figure 4.1: Flow chart for merging multiple views.

4.1 Similarity Measures

The merging process needs to determine which features are the same and which are not, so that the new features can be added into the *b-rep* description model in the global frame. The similarity of two features (vertices, edges, faces, etc.) can be verified based on the distance between their corresponding vertices. Since all the features lie in the same global frame, if two features are the same, they must lie close to each other, i.e., the distance between them should be close to zero.

In practice, however, a threshold should be set interactively to make the similarity verification method more robust when noise and calculation errors exist. Note that this threshold is influenced by the scale and arrangement of objects, especially by the minimum distance between two different vertices.

Similarity of Two Vertices

Let ${}^{a}P$ be the point transferred from local frame A, ${}^{b}P$ be the point transferred from local frame B. If the distance $D_{{}^{a}P,{}^{b}P}$ between vertices ${}^{a}P$ and ${}^{b}P$ is less than a threshold δ,

$D_{{}^{a}P,{}^{b}P} \leq \delta, \quad where \quad \delta \geq 0,$ these two vertices are considered to be the same.

Since the distance between the two nearest points on an object affects the threshold, we can somehow use the minimum distance of two vertices on an object to estimate the upper-bound of δ. The threshold should be less than half of the minimum distance between vertices. Otherwise, a transferred point could probably be misclassified. The knowledge of the noise process and calculation errors can also be helpful in evaluating δ here.

Similarity of Two Edges

Let ${}^aE$ be an edge transferred from local frame A, ${}^bE$ be an edge transferred from local frame B. The similarity measurement between the edges ${}^aE$ and ${}^bE$ is defined as

$$D_{{}^aE,{}^bE} = \min \{ \frac{D_{{}^aP_1,{}^bP_1} + D_{{}^aP_2,{}^bP_2}}{2} \ , \ \frac{D_{{}^aP_2,{}^bP_1} + D_{{}^aP_1,{}^bP_2}}{2} \}$$

where ${}^aP_1$ and ${}^aP_2$ are the end points of edge ${}^aE$, ${}^bP_1$ and ${}^bP_2$ are the end points of edge ${}^bE$. Since the end points of an edge are labeled arbitrarily, we use the smaller average distance between the two possible combinations.

The two edges are considered to be the same, if $D_{{}^aE,{}^bE} \leq \delta_e$, where δ_e is a threshold and $\delta_e \geq 0$.

Similarity of Two Faces

To measure the similarity of two faces, we use the averaged point distance (APD) between the two faces.

Let the vertices of a face ${}^aF$ transferred from frame A be ${}^aP_i$, $i = 0 \ \ to \ \ N-1$, and the vertices of a face ${}^bF$ transferred from frame B be ${}^bP_i$, $i = 0 \ \ to \ \ N-1$. The APD between these two faces is defined as

$$D_{{}^aF,{}^bF} = \min_i \{ \frac{1}{N} \sum_{j=0}^{N-1} D_{{}^aP_j,{}^bP_k} \}$$

where $i = 0, 1, ...N-1$ and $k = i + j \ \ Mod \ N$.

Note that in the face model, a face is specified by a sequence of consecutive bounding vertices, while the start point is arbitrarily chosen among the vertices in

this sequence. Hence, to calculate the APD of two faces, we only need to check those combinations generated by cyclically shifting the sequence of the bounding vertices of one face. This measurement may still be valid in the case where one face is partially visible. When a fairly small portion of the vertex sequence of a face is missing, we can shift the remaining portion of this vertex sequence and compare it with the whole vertex sequence of another face.

If the APD between two faces is less than a threshold δ_f ($\delta_f \geq 0$), $D_{a_F, b_F} \leq \pm\delta_f$, these two faces are the same.

Note that the APD is based on a set of distances between points, so that even if one of the distances is fairly large (compared with the minimum distance between vertices on an object), the resulting the APD may still fall within the range of δ_f.

Detection of Identical Vertices

In practice, we use the following two-step method to detect identical vertices: 1) Find the identical faces according to APD face similarity measurement. 2) Within each pair of identical faces, find identical vertices based on the similarity measurement of two vertices.

This two-step method is helpful especially when some vertices of different faces are close to each other. For example, in Figure 4.2, the distance between the vertex P_a of the face F_1 and the vertex P_b of the face F_2 is very small. F_3 (dotted-line) is a face transferred from another view. If the identity checking is directly based on the distance between vertices, the vertex P_c of face F_3 could be easily mismatched with the vertex P_a. However, when using the APD first, we find that F_2 and F_3 are the same. Then, among the vertices of F_2 and F_3, the vertex P_c can be correctly

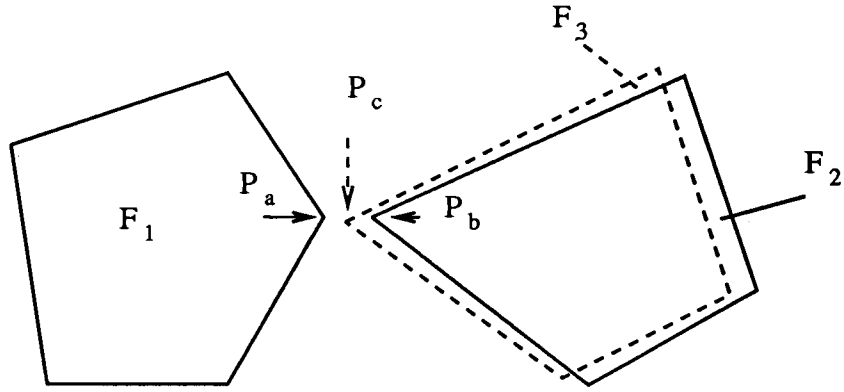


Figure 4.2: Example of checking the identical vertices.

matched with the vertex P_b. Therefore, the above method is more robust than the one which only checks the similarity of isolated vertices.

4.2 Multiple View Integration

Using the generated transformation matrix between views, we transfer two different local views into a global frame so that the the combined description has more information than either of the two local views contains. This newly generated description is successively merged with another view so that the resulting description is closer to a complete one. This process continues until a termination criterion is met.

4.2.1 Termination Criterion

As assumeed at the beginning that the objects involved in this research are closed, bounded, and regular polyhedra [45]. The object with a dangling face, an opening on a surface, or self-intersecting face is invalid. However, in each local view, the object is partially described. In a range image, a jump edge often exists to which only one face is found and the other one is "missing". During the process of integrating such local views, incomplete parts, such as dangling faces and openings on the surface, do possibly exist in the intermediate result. In such a case, the integration process has to be continued. On the other hand, as soon as all the "missing" information is found and the complete description is acquired, the integrating process should stop automatically. Hence, a convenient and effective termination criterion is a must.

> **Termination Criterion:** If every face has a closed cycle of bounding edges and every edge is shared by two such faces, then the merging process can be terminated, otherwise, one or more additional views are required to complete the process.

When this termination criteria is met, the integration procedure should stop and a complete description of polyhedral objects should be obtained.

> If the above criterion holds, then it implies that all the faces are connected with each other. Because the boundary of a face consists of connected edges and each edge attaches to two faces, one can start at any point on one face, go in any direction, cross the boundary of this face and arrive on another face. For the same reason, it is possible to

get from every face to every other face by crossing edges formed by these faces, as long as these faces form one polyhedron. It is also clear that under this criterion, it is not possible for a dangling face to exist because a dangling face must have some edges to which only one face is attached.

Furthermore, if the criterion holds, all the faces will construct a closed surface without any openings. Assume that the criterion holds and yet there exists an opening on the surface. From this it follows that there exist edges to which only one face is attached (e.g. edges which constitute the boundary of the opening). However, this conclusion conflicts with the condition given.

The above discussion indicates that if the termination criterion holds, then the integrated faces of the object in question are connected to each other and the surface of this object is closed. There exist no dangling faces and openings on the surface of the object. To put it another way, the description finally generated under the termination criterion is the complete description of a valid polyhedral object.

4.2.2 Data Structure

In each view, the visible parts of objects are represented in a data structure called *partial-view-list*, which contains the geometry of vertices, edges and faces of the visible part. In addition, the data structure also captures the geometrical and topological relationships among these features. The following shows the main tables in the *partial-view-list.*

Each vertex, edge and face is labeled and assigned a unique serial number. All the tables in the partial-view-list are indexed by these serial numbers.

In Table *f_v*, vertices are ordered in a sequence so that face normal can be calculated according to the right-handed screwing rule.

In table *e_f*, the relationship between an edge and faces is stored. It tells which two faces attach to an edge. Also, it indicates which attached face is the left-handed face and which is the right-handed face, after the start point and end point of the edge being specified.

In each of the tables mentioned above, there is a status flag SF which indicates whether the geometric feature (edge, face, etc.) is complete or not. For instance, when the status flag in table *f_v* is set to one, $SF = 1$, it means that some of the bounding edges of the face are missing. The status flags are very useful in dealing with the partially visible faces or edges.

Of course, there is redundant information in the above tables and we can further compress it and represent this information in a more compact format such as Winged-edge representation. However, we keep this explicit representation, because

Table 4.1: Tables in the partial-view-list.

Table *vertex_T* (coordinates of visible vertices)

Vertex Label	X Coordinate	Y Coordinate	Z Coordinate

Table *e_f* (two faces attached to an edge)

Edge Label	Start Point	End Point	L-Face	R-Face

Table *f_v* (vertices in a face)

Face Label	Vertex 1	Vertex 2	Vertex 3	.....	Vertex N

Table *o_f* (faces in an object)

Object Label	Face 1	Face 2	Face 3	.....	Face N

it directly supplies required relationships among vertices, edges and faces, rather than requiring us to derive this information. Since during the merge and matching procedures, the relationships among vertices, edges and faces are frequently requested, using this type of explicit representation saves computation time.

For each view, a set of look up tables is used to remember the corresponding information between views. These look-up tables significantly reduce the computation cost in the merging process.

The data structure used for the final global description is called the *complete-view-list.* It has the same structure as the partial-view-list, except that is contains a complete description of the whole object.

4.2.3 Updating Description

As indicated at the beginning of this chapter, the purpose of the merging procedure is to use the correspondence between views to link all the information which is described in different views into a global description. Since each local view contains different information, by merging the partial descriptions of local views, we can incrementally add new information to the description in a global frame and finally get the complete description of objects.

Figure 4.3 shows a flow chart of our merge algorithm. The input to this module is the partial descriptions of a 3-dimensional object (the *partial-view-list* for each of the local views). The output of this algorithm is the *completed-view-list* which contains the complete information about the object.

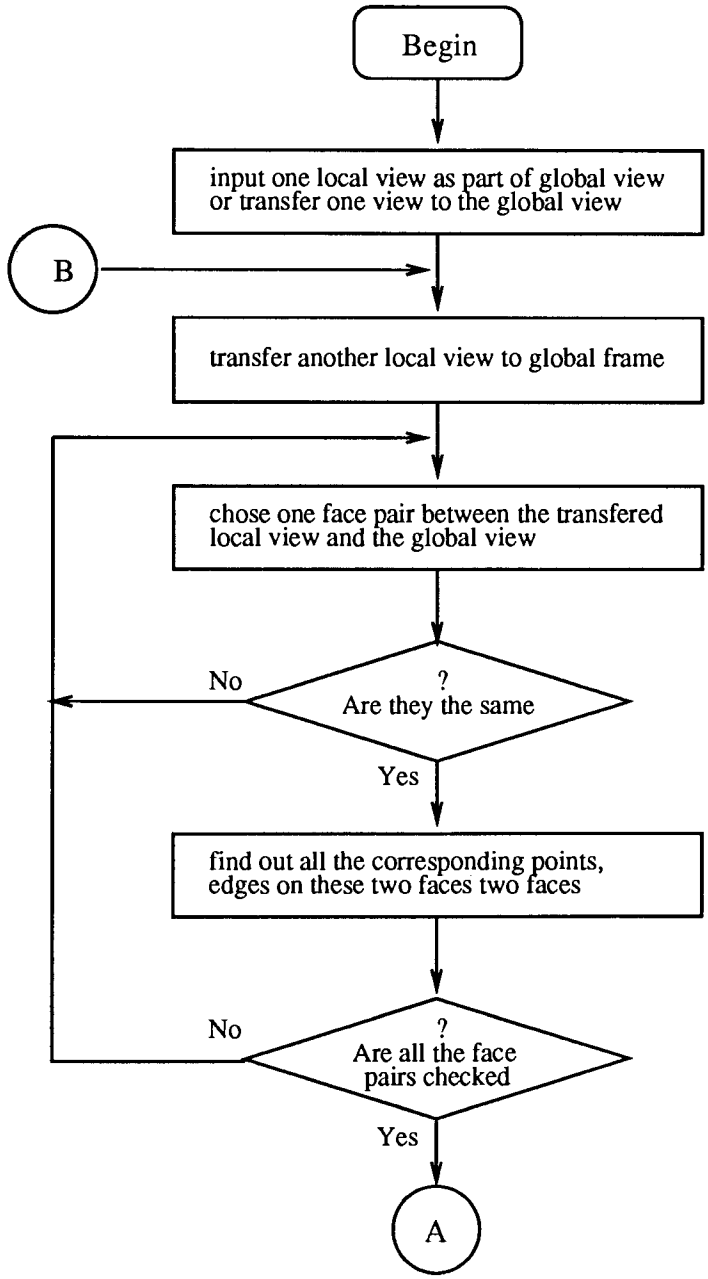


Figure 4.3: Merging — continued

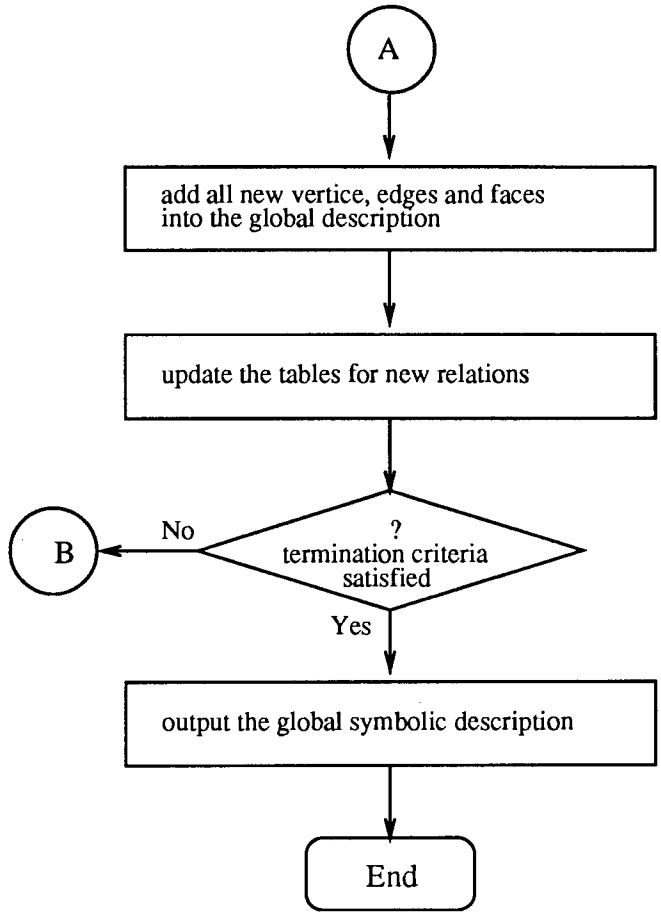


Figure 4.4: Merging

To update the previous global description, first check whether a feature (vertex, edge, face, etc.) transferred from a new local view is present in the global description. If so, simply get rid of the transferred feature, because this feature is already considered in the previous global description. If there is no corresponding feature, then this feature is a new feature which should be added into the global description.

Besides adding the geometry of a new feature into the global description, we also update the previous description by adding all the relationships associated with the new feature, e.g., merging the relationships stored in the e_f tables.

The whole algorithm is summarized in the following steps:

1. Select a global frame which could be the same as one of the local view frames. The transformation between the selected global frame and one of the local frames is known.

2. Transfer the partial description in another local frame into the global frame.

3. In the global frame, check which faces transferred from different views are identical or overlap each other. Based on the result of this checking, find the same vertices and edges transferred from different views and update look-up tables with the information about correspondence.

4. Update the symbolic global description (complete-view-list) by adding the new faces, edges and vertices.

5. Update all the relationships. When two local descriptions are merged, new relationships among face, edge, and vertex, which are not given in either local description, may occur.

6. Check whether or not the termination criterion is met. If this criterion is satisfied, then stop and output the final global description. Otherwise repeat step 2, 3, 4, 5 and 6, until the termination criterion is met, or all the local views are integrated.

CHAPTER 5

Results and Analysis

This chapter reports the experimental results on synthetic range images and provides an analysis of system performance. First, we present results for the noiseless case. Two objects, object I and object II were used in the experiment. Object I was a car and object II was a seven-faced polyhedron. A set of synthetic range images were derived at different viewpoints for each object. The size of the range images is 150×150, the depth data were represented in 8 bits. The distance between the camera and the object is around 100 units. Both sets of range images satisfied the assumptions that there exist at least two adjacent faces common in two views and that each face of the object exists in at least one view. Note that there were many symmetric features in each of the test objects that challenged our matching algorithm. Since we assumed that face models of visible surfaces of objects from low level image processing were available, in our experiments these face models were manually entered. The XYZ coordinates of vertices were represented as real numbers. The hierarchical matching algorithm was first applied to obtain the corresponding point pairs between views.

Then, based on this, the solid-body transformations between views were generated and refined if necessary. Finally, all the local views were transferred into a global frame and all the local descriptions were consolidated and a complete 3-dimensional *b-rep* description was created as the output. Both the objects were successfully handled by our system. Next, we tested our system on synthetic ranges with noise added. The results are discussed in Section 5.4

5.1 Object I and II

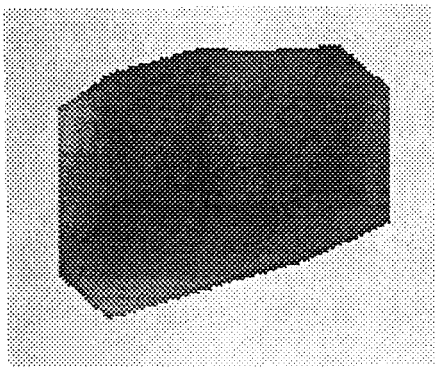
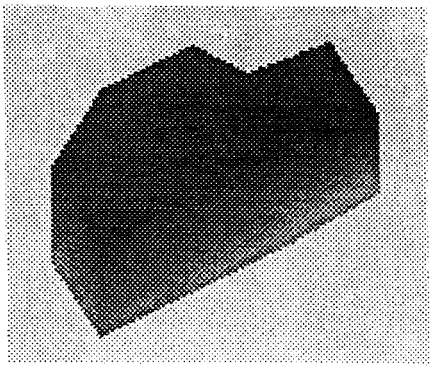
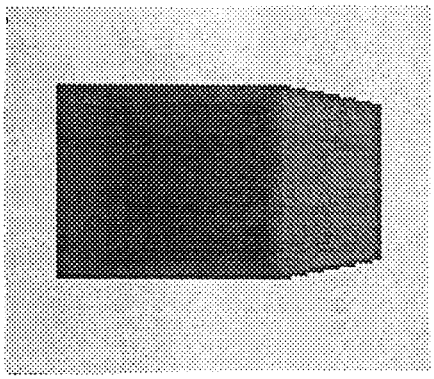
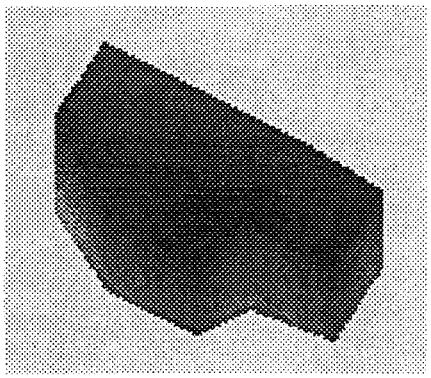
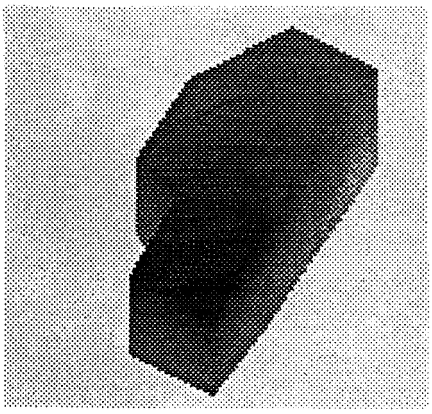
The object car (see Figure 1.2), had seventeen vertices, twenty-seven edges and thirteen faces. It had many symmetric local parts which challenged our system.

Five range images, shown in Figure 5.1, were taken from different viewpoints to cover the whole object. The face model of the visible surface of the object car in each view [1] is shown in Figure 5.2. The *partial-view-list* for the object car in each local view is shown in Table 5.1 and Table 5.2.

As described in Section 4.2.2, each vertex, edge, or face was assigned a serial number respectively. The physical positions and relationships are given in the form of a set of tables. Vertices of a face are listed cyclically in the counter-clockwise direction. For instance, the XYZ coordinate of vertex 11 (in Table *vertex_T*) in this local view is (84.26, 70.98, 92.54) [2]. Face 1 (in Table *f_v*) is specified by the

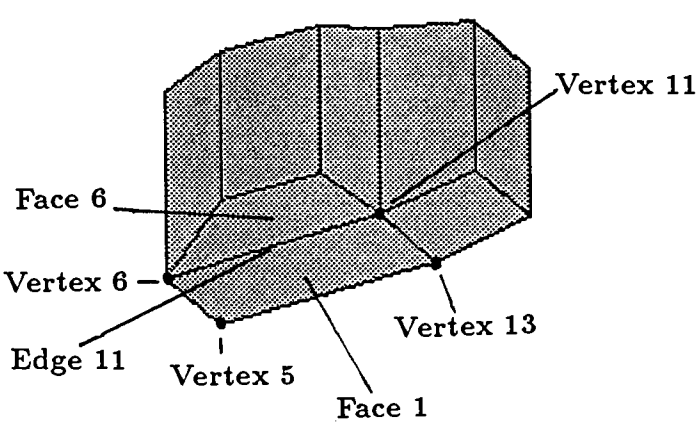
[1] This is given, by assumption, from low level image processing.

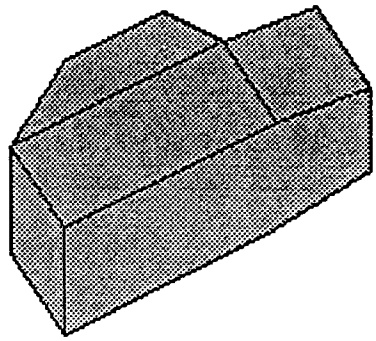
[2] The numbers are rounded for display.

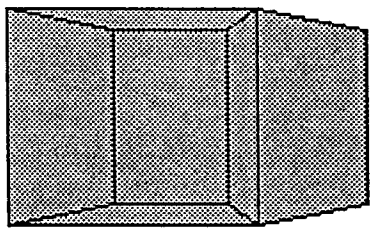






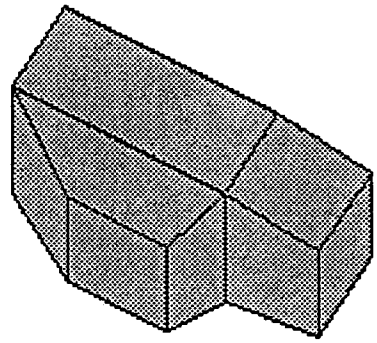
car_view1	car_view2
car_view3	car_view4
car_view5	

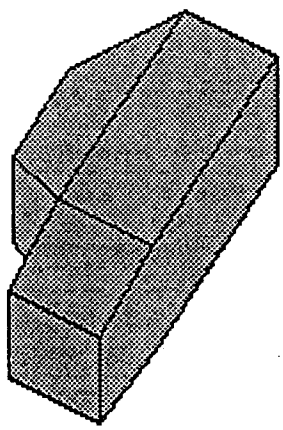
Figure 5.1: Multiple range images of object I.





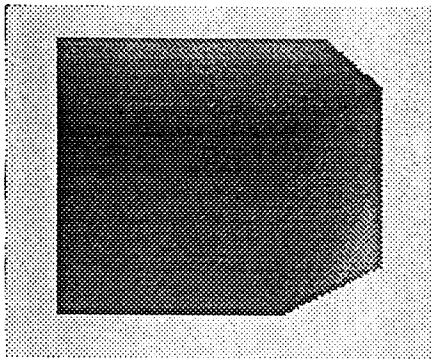
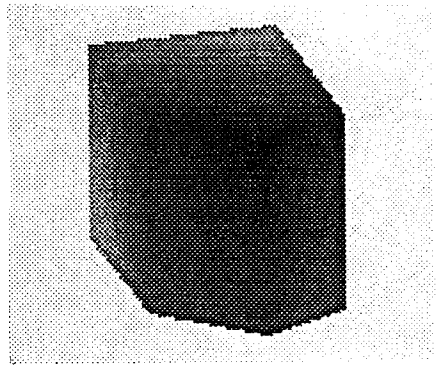
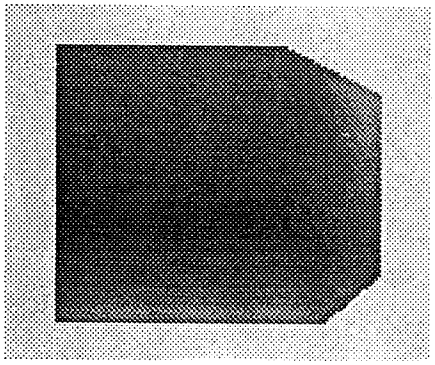
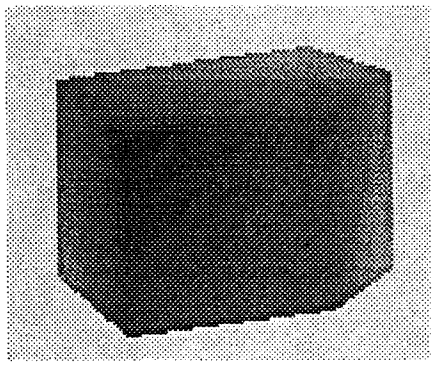
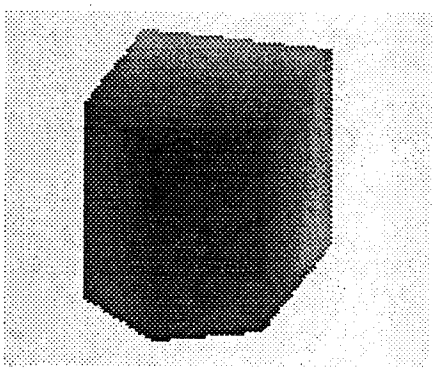






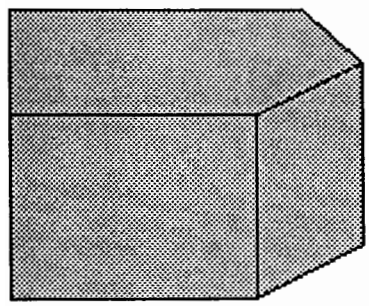
car_view1	car_view2
car_view3	car_view4
car_view5	

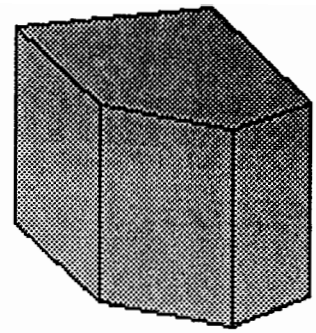
Figure 5.2: Face model of visible surface of object I.

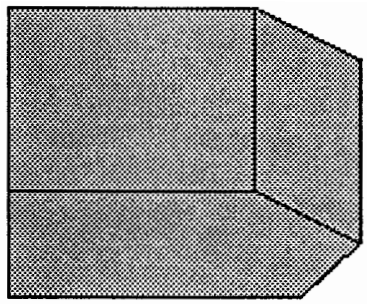






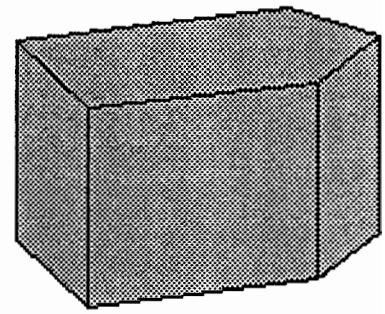
rock_view1	rock_view2
rock_view3	rock_view4
rock_view5	

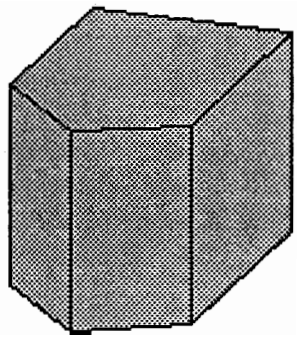
Figure 5.3: Multiple range images of object II.











rock_view1	rock_view2
rock_view3	rock_view4
rock_view5	

Figure 5.4: Face model of visible surface of object II.

Table 5.1: Car_view2 (view 1 in object car) — continued.

vertex_T (XYZ Coordinates of vertices)

Vertex Label	X	Y	Z
1	42.99	30.00	129.05
2	32.30	70.98	122.54
3	30.00	44.99	106.57
4	49.80	131.60	152.85
5	114.75	116.60	145.35
6	101.76	131.60	122.85
7	80.40	115.62	97.86
8	38.83	115.62	121.86
9	72.40	87.90	84.01
10	30.83	87.90	108.01
11	84.26	70.98	92.54
12	71.56	44.99	82.55
13	97.25	55.98	115.04
14	84.55	30.00	105.05

Num_VEFOS (The number of geometric features in the view)

Vertex	Edge	Face	Object
14	21	8	1

Table 5.2: Car_view2 (view1 in object car).

e_f (The two faces attached to an edge)

Edge Label	Start Point	End Point	L-Face	R-Face
1	2	3	4	0
2	3	1	3	0
3	5	6	1	0
4	6	4	5	0
5	6	7	6	5
6	7	8	8	5
7	8	4	0	5
8	8	10	8	0
9	10	2	7	0
10	7	9	6	8
11	6	11	1	6
12	9	11	6	7
13	9	10	7	8
14	2	11	7	4
15	11	12	2	4
16	12	3	3	4
17	12	14	2	3
18	1	14	3	0
19	13	11	2	1
20	14	13	2	0
21	5	13	0	1

f_v (vertices in a face)

Face Label	Vertex 1	Vertex 2	Vertex 3	Vertex 4
1	5	13	11	6
2	11	13	14	12
3	1	3	12	14
4	3	2	11	12
5	4	6	7	8
6	6	11	9	7
7	2	10	9	11
8	8	7	9	10

cyclical vertex sequence: vertex 5, vertex 13, vertex 11, vertex 6 and vertex 5 (see Figure 5.2). The right-handed face and the left-handed face of edge 11 (in Table *e_f*) are face 1 and face 6 respectively. Face 0 in the table that means the face has not been detected.

Test object II, which had 10 vertices, 15 edges, and 7 faces, was a more complex polyhedron which contrasted with the highly symmetric object car. Five views and the corresponding face models of visible surfaces are shown in Figure 5.3 and Figure 5.4.

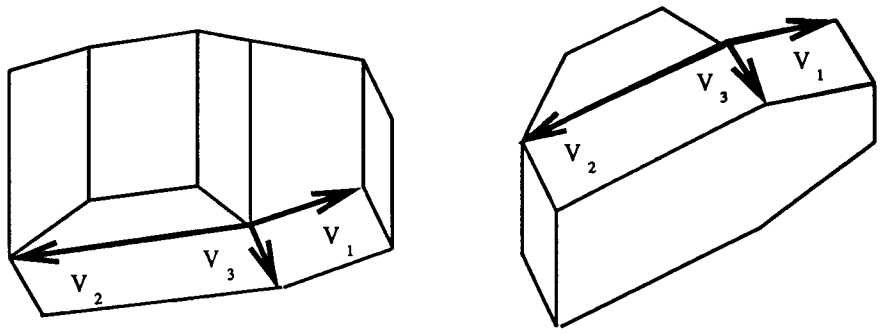
The number of local views needed is influenced by different selection of viewpoints. For example, the complete *b-rep* description for the object car can be obtained by integrating either view 1, view2, view 3, and view4 or view3, view4 and view5. This fact reveals the necessity of optimally chosing viewpoints with an active vision mechanism — a widely open issue (see Chapter 6).

5.2 Performance of Matching Algorithm

This section presents the performance of the hierarchical matching algorithm. A set of triple branch structures is first extracted from the face models of visible surfaces of the object in each local view. Then, the hierarchical search strategy is applied and different constraints are applied to reduce the computation time for matching these features between views.

Triple Structures

Examples of the triple branch structures in the local views, Car_view1 and



View 1 View 2

Figure 5.5: A triple pair.

Car_view2, are shown in Tables 5.3 and 5.4 respectively. There are 22 triple branch structures in local view Car_view1 and 12 in Car_view2. The number of triple features could be doubled, if the feature is defined improperly (see the discussion in Chapter 3). The invariance property of the triple branch structure reduces the number of possible triple pairs by a factor of two (compared with the case where the two subordinate branches are arbitrarily labeled).

Geometrical Constraints

Instead of spending equal computation costs (checking six constraints, three angles and three lengths) on every triple pair, we distribute these six constraints at different search levels and let those triple pairs which are more likely to be correct matching pairs receive more attention.

An example is shown in Table 5.2. The number in each column indicates how many triple pairs have satisfied the constraint shown at the top of the column. It is clear that only a small number of triple pairs need to be checked by all the six

Table 5.3: Triple for Car_view1.

Triple	*vector 1*		*vector 2*		*vector 3*		*Left-handed*	*right-handed*
Label	*V1S*	*V1E*	*V2S*	*V2E*	*V3S*	*V3E*	*face Label*	*face Label*
1	6	4	6	11	6	7	6	5
2	7	6	7	9	7	8	8	5
3	7	8	7	6	7	9	6	8
4	6	7	6	5	6	11	1	6
5	9	10	9	7	9	11	6	7
6	9	7	9	11	9	10	7	8
7	2	3	2	10	2	11	7	4
8	11	2	11	13	11	12	2	4
9	12	11	12	14	12	3	3	4
10	12	3	12	11	12	14	2	3
11	13	5	13	14	13	11	2	1
12	7	9	7	8	7	6	5	6
13	8	10	8	4	8	7	5	8
14	9	11	9	10	9	7	8	6
15	11	13	11	9	11	6	6	1
16	11	6	11	2	11	9	7	6
17	10	2	10	8	10	9	8	7
18	11	9	11	12	11	2	4	7
19	12	14	12	3	12	11	4	2
20	3	1	3	2	3	12	4	3
21	14	13	14	1	14	12	3	2
22	11	12	11	6	11	13	1	2

V1S, V2S, and V3S are the start point of the vectors.
V1E, V2E, and V3E are the end point of the vectors.

Table 5.4: Triple for Car_view2.

Triple Label	*vector 1*		*vector 2*		*vector 3*		*Left-handed face Label*	*right-handed face Label*
	V1S	*V1E*	*V2S*	*V2E*	*V3S*	*V3E*		
1	3	4	3	2	3	5	5	1
2	5	3	5	11	5	6	2	1
3	6	7	6	5	6	9	2	4
4	11	5	11	12	11	9	3	2
5	12	1	12	10	12	11	3	5
6	5	6	5	3	5	11	5	2
7	5	11	5	6	5	3	1	5
8	6	9	6	4	6	5	1	2
9	9	11	9	8	9	6	4	2
10	9	10	9	6	9	11	2	3
11	11	9	11	5	11	12	5	3
12	11	12	11	9	11	5	2	5

V1S, V2S, and V3S are the start point of the vectors.

V1E, V2E, and V3E are the end point of the vectors.

Table 5.5: Number of triple pairs passed at different searching levels.

View A	View B	Triple Pairs	Same Principle	Same Face Shape	Same Magnitude
View0	View1	484	76	30	22
View2	View1	264	40	7	4
View3	View1 +View2	612	68	14	12
View4	View1 +View2 +View3	880	140	31	21
View3	View4	396	68	14	12
View1	View4	484	76	20	12
View2	View1 +View4	432	64	12	7
Average		507	76	18	13

constraints. The threshold for comparing if two angles are equal was set at 5^0. The threshold for checking if two vectors have equal magnitudes was 4 units.

Priority Queue

Instead of verifying all the candidate triple pairs, our algorithm sorts all the candidates into a priority queue to find those triple pairs which are most likely to represent the correct match between the views. First, each triple pair votes for its

		View 1														
	Vertex	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
V	5	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0
i	6	0	0	0	0	0	3	0	0	0	0	0	0	0	1	0
e	7	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
w	8	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2	9	0	0	0	0	0	0	0	0	0	0	4	0	2	0	0
	10	0	0	0	0	1	0	0	0	0	0	0	1	0	1	0
	11	0	0	1	0	0	0	0	0	0	0	2	0	3	0	0
	12	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0
	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.6: Score of vertex pairs for Car_view1 and Car_view2.

vertex pairs. Figure 5.6 shows the voting result between local views Car_view1 and Car_view2. The potentially matched vertex pairs have received more votes, e.g. the score of vertex pair $(11, 9)$ equals 4.

Table 5.6: Scored triple pairs for Car_view1 and Car_view2.

Triple Pair Label	*View I*						*View II*						Score
	vector 1		*vector 2*		*vector 3*		*vector 1*		*vector 2*		*vector 3*		
	V1S	V1E	V2S	V2E	V3S	V3E	V1S	V1E	V2S	V2E	V3S	V3E	
1	6	7	6	5	6	11	6	7	6	5	6	9	10
2	13	5	13	14	13	11	11	5	11	2	11	9	10
3	11	12	11	6	11	13	11	5	11	2	11	9	6
4	12	11	12	14	12	3	12	1	12	10	12	11	4
5	11	13	11	9	11	6	9	11	9	8	9	6	11
6	13	5	13	14	13	11	9	10	9	6	9	11	6
7	11	12	11	6	11	13	9	10	9	6	9	11	11

Then, each of the triple pairs is graded with the score of its vertex pairs. Table 5.6 shows the triple pairs and their scores. The highest score is achieved only when all the triples in this group find the correct matches. With such a prioritizing process, we implicitly group several locally connected triples together and check the correct match based on such groups instead of individual triples.

The experimental results indicate, as shown in table 5.7, table 5.8 and table 5.9, that the triple pairs with the highest score almost always appear to be the correct match. When the scores of triple pairs decrease, the chances of mismatching increase. The third column of the tables indicates whether a triple pair stands for the correct match between views. According to this fact, the triple pairs should be chosen for

verification in order from the highest score of these triple pairs to the lowest one. It is also possible to test only a few candidate triple pairs with the highest scores.

Table 5.7: Triple pair score versus correct match for Car_view1trans and Car_view2.

Triple Pair Score	Triple Pair Label	Correct Match ?
40	5	Yes
40	6	Yes
40	17	Yes
39	19	Yes
39	22	Yes
38	18	Yes
36	30	Yes
34	1	Yes
34	4	Yes
34	9	Yes
33	10	Yes
33	11	Yes
33	23	Yes
32	2	Yes
32	3	Yes
32	7	Yes
32	15	Yes
27	13	Yes
27	20	Yes
22	16	Yes
21	26	Yes
20	28	Yes
6	8	No
6	12	No
6	14	No
6	21	No
6	27	No
6	29	No
4	24	No
4	25	No

Verification

The next step is to verify whether the triple pairs chosen can generate the correct transformation between views. This is the most time-consuming procedure of all.

However, after the previous search, the number of candidate triple pairs left for this test is already much smaller compared with the number of original triple pairs.

Each candidate triple pair is used to generate a transformation. Then, the points in one view are transferred to another view with this transformation. The verification is based on the number of points which are transferred from the first view and overlap with the points in the second view. It is usual that the correct transformation can be satisfied by more points between views than the incorrect one can. All the correct corresponding point pairs should satisfy the correct transformation and the different incorrect corresponding point pairs are less likely to satisfy one transformation. Table 5.12, Table 5.10, Table 5.11, Table 5.13, Table 5.14, and Table 5.15 show the scores of the possible transformations generated by candidate triple pairs. The correctness of the transformations is also indicated. From these tables, it is obvious that the correct transformation has always received the largest score.

5.3 Multiple View Integration

After finding the transformations between views, we can integrate the local views one by one. The threshold used to check two identical vertices was 4. Table 5.16 and Table 5.17 give the intermediate results of the integration process. During the integration, the number of vertices, edges and faces increases, as the number of local views integrated into the global view increases. The bold letters in the tables indicate the recent change. It is shown that in *e_f*, the 0's in the columns named right-handed face and left-handed face are gradually reduced. In the end, each edge has two supporting faces, i.e., the criteria of termination of integration are met so that the description of the object in question is completed. Table 5.18 and Table 5.19 show the final complete *b-rep* description.

5.4 Test with Noisy Data

To simulate more realistic situations, we also tested our system with noise added to synthetic range images. Noise may arise due to a variety of sources, e.g. calibration and low level image segmentation errors. We added different levels of noise (2%, 5%, 8% and 10%, respectively of the maximum range values) to the coordinates of vertices in the face models as shown in Figure 5.7. To show the noisy data clearly, we have superimposed all the views in one frame, Figure 5.8. Since the local views were generated by a synthetic range image generator, we knew the actual transformations between views, so that we can transform all the local views into one frame with these transformations. It is clear that as the noise level increases, the distortion of the geometrical features is more severe.

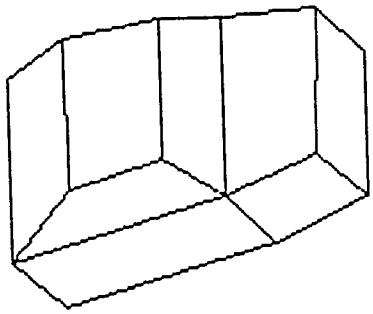
We tested our system with different levels of noise. The whole system worked well with the data of 2% and 5% noise. The thresholds used in both cases were within a reasonable range. We found a set of thresholds which could be applied to both cases (the thresholds for checking the angles and magnitudes of vectors were 19^0 and 11 units respectively). Basically, the minimum distance between two vertices and the presence of similar local geometrical structures are two important factors that affect the thresholds and hence the system performance. Note that there are many symmetric features in the test data. For instance, the left-side and right-side of the car are symmetric. As expected, when the noise became large, the thresholds needed to be increased. With the noise level at 8%, some errors began to occur. To further analyze the performance of our system in some detail, we tested the matching module, the process of transformation estimation, and the merging module separately. From the test results, we found that the most sensitive part of the system is the process of transformation estimation. With the noise level less than or equal to 8%, the matching module and merging module still generated correct results, but the estimated transformations were quite different from the actual transformations. However, with the noise level larger than 8%, the matching module and merging module also started to give wrong results. For instance, some points should have been considered to be identical, but were not.

The estimated transformation is quite sensitive to noise in the coordinates of corresponding vertex pairs. A small error in the position of vertices may affect the estimated transformation significantly. This is true especially when the number of corresponding point pairs is small. This error is then propagated to the merging module and causes errors in the process of intergration. We illustrate this in Figure 5.9. Note the difference in superimposed views in Figure 5.9.A (obtained by

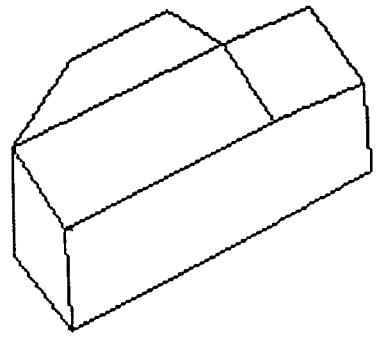
using actual transformations) and Figure 5.9.B (obtained by using estimated transformations). Clearly, the estimated transformations are much different from the actual ones.

Hence, we can say that relatively larger errors are generated in the process of estimating a rigid-body transformation. However, when more corresponding point pairs were used to estimate transformations, errors in generated transformations were reduced. From our results on noisy synthetic range images, we can see that it is important to take more corresponding point pairs or get relatively noise-free face models in local views.

Although we don't have a range scanner in our lab to perform real experiments, most range scanners can easily provide an accuracy of 1 part in 1000 [7]. This corresponds to a 0.1% error, which is much less than the noise limits within which our system performs successfully. However, we are cautious since there are other sources of errors as well, e.g. low-level segmentation. Note also that we are interested in building models for planning collision-free paths and therefore don't require very strict accuracy as in the case of some inspection tasks.

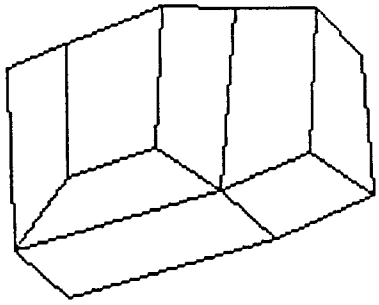


View 1

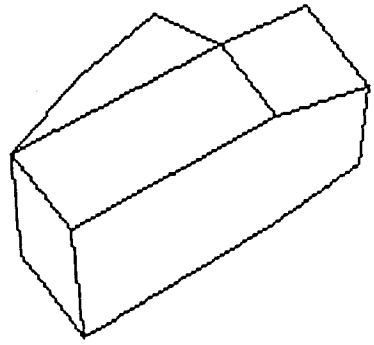


View 2

2% noise data

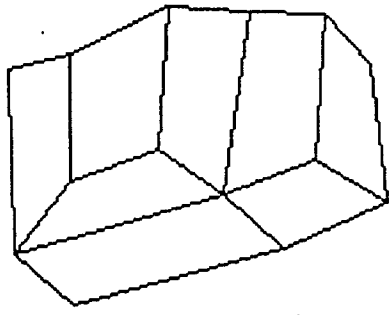


View 1

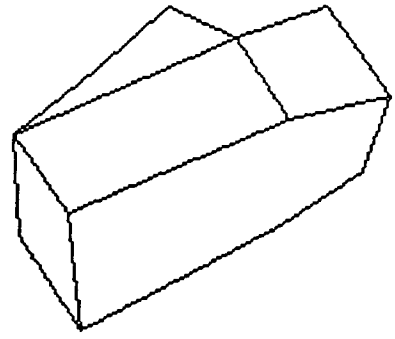


View 2

5% noise data



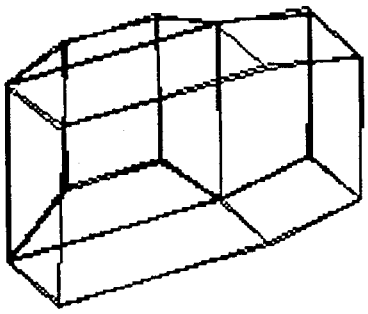
View 1



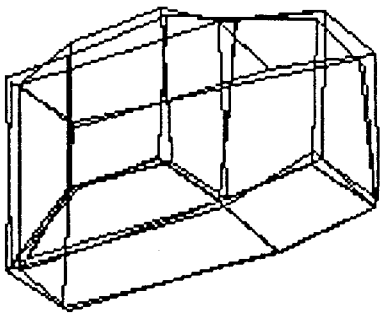
View 2

8% noise data

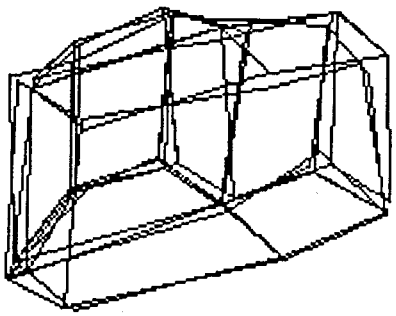
Figure 5.7: Noisy Data



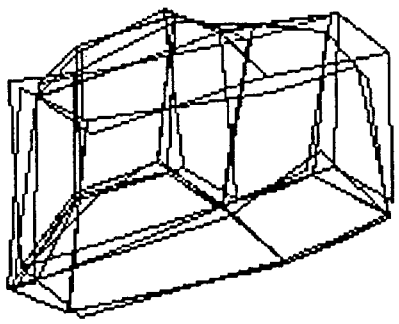
2% noise data



5% noise data

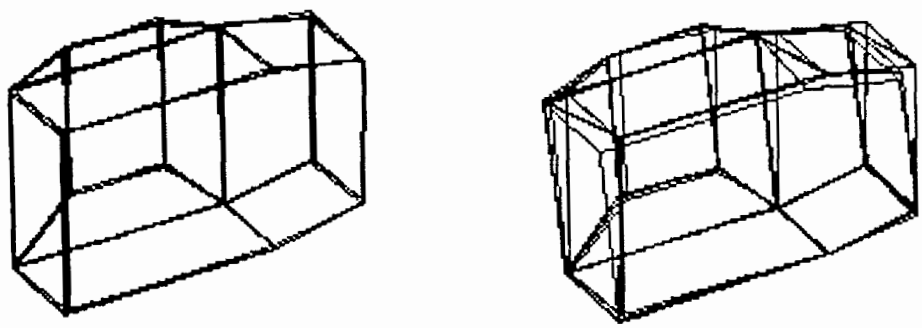


8% noise data



10% noise data

Figure 5.8: Superimposed Noisy Views



(A) Actual Transformation (B) Estimated Transformation

Figure 5.9: 2% noise-level: Different views are transformed in one frame and superimposed. Actual transformations were used in (A) whereas estimated transformations were used in (B). This indicates that the estimated transformations were quite different than the actual ones.

Table 5.8: Triple pair score versus correct match for Car_view4+Car_view1 and Car_view2.

Triple Pair Score	Triple Pair Label	Correct Match ?
11	8	Yes
11	11	Yes
10	1	Yes
10	2	Yes
8	5	No
8	7	No
8	10	No
6	3	No
6	4	No
6	9	No
6	12	No
4	6	No

Table 5.9: Triple pair score versus correct match for Car_view1 and Car_view2.

Triple Pair Score	Triple Pair Label	Correct Match ?
11	5	Yes
11	7	Yes
10	1	Yes
10	2	Yes
6	3	No
6	6	No
4	4	No

Table 5.10: Score of transformations versus correct match for Car_view4 and Car_view1.

Score of transformation	Triple Pair Label	Correct Match ?
12	3	Yes
12	4	Yes
12	5	Yes
12	6	Yes
12	9	Yes
12	10	Yes
12	11	Yes
12	12	Yes
8	2	No
8	4	No
8	10	No
8	13	No
7	1	No
7	11	No
6	3	No
6	8	No
6	12	No
6	14	No
6	17	No
6	19	No

Table 5.11: Score of transformations versus correct match for Car_view1 and Car_view2.

Score of transformation	Triple Pair Label	Correct Match ?
10	1	Yes
10	2	Yes
10	5	Yes
10	7	Yes
6	3	No
6	6	No
5	4	No

Table 5.12: Score of transformations versus correct match for Car_view1trans and Car_view2.

Score of transformation	Triple Pair Label	Correct Match ?
14	1	Yes
14	2	Yes
14	3	Yes
14	4	Yes
14	5	Yes
14	6	Yes
14	7	Yes
14	9	Yes
14	10	Yes
14	11	Yes
14	13	Yes
14	15	Yes
14	16	Yes
14	17	Yes
14	18	Yes
14	19	Yes
14	20	Yes
14	22	Yes
14	23	Yes
14	26	Yes
14	28	Yes
14	30	Yes
6	8	No
6	12	No
6	14	No
6	21	No
6	27	No
6	29	No
5	24	No
5	25	No

Table 5.13: Score of transformations versus correct match for Car_view4+Car_view1 and Car_view2.

Score of transformation	Triple Pair Label	Correct Match ?
12	1	Yes
12	2	Yes
12	8	Yes
12	11	Yes
10	5	No
10	7	No
10	10	No
7	3	No
7	12	No
6	4	No
6	9	No
5	6	No

Table 5.14: Score of transformations versus correct match for Car_view3 and Car_view1+Car_view2.

Score of transformation	Triple Pair Label	Correct Match ?
10	1	Yes
10	2	Yes
10	3	Yes
10	4	Yes
10	5	Yes
10	6	Yes
10	8	Yes
10	9	Yes
10	10	Yes
10	11	Yes
10	12	Yes
10	14	Yes
6	7	No
6	13	No

Table 5.15: Score of transformations versus correct match for Car_view4 and Car_view3+Car_view1+Car_view2.

Score of transformation	Triple Pair Label	Correct Match ?
14	5	Yes
14	6	Yes
14	7	Yes
14	8	Yes
14	9	Yes
14	10	Yes
14	12	Yes
14	22	Yes
14	23	Yes
14	24	Yes
14	25	Yes
14	26	Yes
14	28	Yes
14	30	Yes
10	2	No
10	4	No
10	13	No
10	15	No
10	19	No
10	20	No
10	31	No
7	1	No
7	16	No
6	3	No
6	11	No
6	17	No
6	18	No
6	21	No
6	27	No
6	29	No
5	14	No

Table 5.16: After integration of Car_view1 and Car_view2.

e_f (The two faces attached to an edge)

Edge Label	Start Point	End Point	L-Face	R-Face
1	2	3	4	0
2	3	1	3	0
3	5	6	1	**9**
4	6	4	5	**9**
5	6	7	6	5
6	7	8	8	5
7	8	4	0	5
8	8	10	8	0
9	10	2	7	0
10	7	9	6	8
11	6	11	1	6
12	9	11	6	7
13	9	10	7	8
14	2	11	7	4
15	11	12	2	4
16	12	3	3	4
17	12	14	2	3
18	1	14	3	**10**
19	13	11	2	1
20	14	13	2	**10**
21	5	13	**10**	1
22	1	**15**	**10**	0
23	**15**	**16**	**10**	0
24	**16**	4	**9**	0
25	**16**	5	**10**	**9**

Num_VEFOS (The number of geometric features in the view)

Vertex	Edge	Face	Object
16	**25**	**10**	1

Table 5.17: After integration of Car_view3 and Car_view1+Car_view2.

e_f (The two faces attached to an edge)

Edge Label	Start Point	End Point	L-Face	R-Face
1	2	3	4	0
2	3	1	3	0
3	5	6	1	9
4	6	4	5	9
5	6	7	6	5
6	7	8	8	5
7	8	4	**11**	5
8	8	10	8	**11**
9	10	2	7	**11**
10	7	9	6	8
11	6	11	1	6
12	9	11	6	7
13	9	10	7	8
14	2	11	7	4
15	11	12	2	4
16	12	3	3	4
17	12	14	2	3
18	1	14	3	10
19	13	11	2	1
20	14	13	2	10
21	5	13	10	1
22	1	15	10	0
23	15	16	10	0
24	16	4	9	0
25	16	5	10	9
26	4	2	**11**	0

Num_VEFOS (The number of geometric features in the view)

Vertex	Edge	Face	Object
16	**26**	**11**	1

Table 5.18: Complete *b-rep* description of object car - continued.

e_f (The two faces attached to an edge)

Edge Label	Start Point	End Point	L-Face	R-Face
1	2	3	4	**12**
2	3	1	3	**12**
3	5	6	1	9
4	6	4	5	9
5	6	7	6	5
6	7	8	8	5
7	8	4	11	5
8	8	10	8	11
9	10	2	7	11
10	7	9	6	8
11	6	11	1	6
12	9	11	6	7
13	9	10	7	8
14	2	11	7	4
15	11	12	2	4
16	12	3	3	4
17	12	14	2	3
18	1	14	3	10
29	13	11	2	1
20	14	13	2	10
21	5	13	10	1
22	1	15	10	**12**
23	15	16	10	**13**
24	16	4	9	**13**
25	16	5	10	9
26	4	2	11	**13**
27	15	2	**13**	**12**

Num_VEFOS (The number of geometric features in the view)

Vertex	Edge	Face	Object
16	**27**	**13**	1

Table 5.19: Complete *b-rep* description of object car.

vertex_T (*XYZ* Coordinate of vertices)

Vertex Label	X	Y	Z
1	74.00	10.00	50.00
2	80.00	40.00	80.00
3	74.00	10.00	80.00
4	80.00	110.00	80.00
5	20.00	110.00	50.00
6	20.00	110.00	80.00
7	26.00	80.00	100.00
8	74.00	80.00	100.00
9	26.00	48.00	100.00
10	74.00	48.00	100.00
11	20.00	40.00	80.00
12	26.00	10.00	80.00
13	20.00	40.00	50.00
14	26.00	10.00	50.00
15	80.00	40.00	50.00
16	80.00	110.00	50.00

f_v (vertices in a face)

Face Label	Vertex 1	Vertex 2	Vertex 3	Vertex 4	Vertex 5	Vertex 6
1	5	13	11	6		
2	11	13	14	12		
3	1	3	12	14		
4	3	2	11	12		
5	4	6	7	8		
6	6	11	9	7		
7	2	10	9	11		
8	8	7	9	10		
9	*16*	*5*	*6*	*4*		
10	*1*	*14*	*13*	*5*	*16*	*15*
11	*2*	*4*	*8*	*10*		
12	*1*	*15*	*2*	*3*		
13	*15*	*16*	*4*	*2*		

CHAPTER 6

Conclusion

This thesis has presented an approach for extracting a complete 3-dimensional *b-rep* description of a polyhedral object from multiple range images taken from different viewpoints. No prior knowledge of the transformations relating different viewpoints is assumed. This *b-rep* description is useful for robotic manipulation tasks such as path planning and grasp planning.

Starting from basic face models of visible surfaces of objects [1] in each local view, our system extracts features, matches these features, generates rigid-body transformations that relate the local views, and finally merges these local views into a 3-dimensional *b-rep* description of the object. The main features and points raised in this thesis are summaried below.

1. The triple branch structure is selected as a main matching feature, which

[1] We assume that the basic face models of visible surfaces of objects in each local view are available.

not only helps the efficient matching algorithm, but also suffices to generate rigid-body transformation between views.

2. A hierarchical matching algorithm is designed which sequentially uses intrinsic features at different search stages. A prioritizing procedure is proposed to arrange the order (priority queue) among matching candidates. This matching algorithm is much faster when compared to the brute force search method.

3. A method of integrating multiple views is proposed and demonstrated. It will merge different local views together and then output the complete 3-dimensional *b-rep* description of the polyhedral object in the scene. A convenient and effective termination criterion is set to monitor the whole merging process. This termination criterion also makes it possible to utilize the active vision mechanism later.

4. The minimum number of corresponding point pairs to generate rigid-body transformation between two views is discussed explicitly. It is indicated that at least three non-colinear corresponding point pairs are needed to generate a rigid-body transformation. Also, a condition is defined to confirm that an arbitrary transformation is a rigid-body transformation between two given sets of points.

5. A synthetic range image generator has been designed and implemented. Our system, in conjunction with this synthetic range image generator, forms a useful developing tool in the area of robotics, *CAD*, and geometric modeling.

All the algorithms and methods are implemented mainly in C and MATLAB. The front-end graphics have been implemented with *HOOPS*.

Our research is only an initial step in this promising direction of building *CAD* models from range images. Since no real range data was available, all the range images used in this project were synthetic range images. Although the system performed well with up to 5% noise added to synthetic data, the performance of the system needs to be investigated with real range data to demonstrate the robustness of the system. We anxiously await a laser range scanner to be installed in our lab in September, 1992 to run real experiments.

Although some simple mechanisms are already used in our method to handle the case of partial occlusion, more sophisticated algorithms need to be designed to deal with the case in which a more complicated scene is considered and occlusion becomes the main problem.

Another important point is incorporating certain aspects of active vision within our framework. For instance, how could the next view be selected automatically? Clearly, the requirement that at least two faces be common in two views gives some constraints on the next view. In addition, other geometrical constraints may be useful, e.g. the object surfaces should not be too oblique to the scanner, since range data accuracy suffers. Such criteria need to be investigated.

Another major research direction could be to include curved surfaces, such as ellipsoidal and paraboloidal surfaces, in this multiview integration framework.

Appendix A

A Synthetic Range Image Generator

The synthetic range image generator consists of the following modules: inputting 3-dimensional data, selecting view point, determining face normal, orthogonally projecting visible faces, Z-buffering, and finally generating the range image and the flag image.

(1) Input Data: the input to the synthetic range image generator is a in the form of the *complete-view-list*, in which the faces, edges, and vertices of the object are specified. The view point and the object is located in a global (reference) coordinate system.

(2) Transformation between the object-centered system to view-point centered system: this transformation is used to describe the object in view-point centered system. The view-point centered system is defined with its origin overlapping the

view point and the positive direction of Z-axis aligning with the view vector.

(3) Determination of possible visible faces: in the viewpoint-centered frame, the face normal (pointing outward) of each face of the object is determined and used to detect the possible visible faces. A face is a possible visible face in this frame, if its normal and the Z-axis vector constitute the angle larger than 90 degree.

(4) Orthogonal projection: the 2-dimensional projection of each of the possible visible faces is formed by orthogonally projecting the face to the xy-plane.

(5) Calculation of range: the xy-plane is scanned. If a scan point is inside of the 2-dimensional projection of one face, it is necessary to calculate the distance between this point (which on the xy-plane) and the corresponding surface point (which is on the face of the object and has the same x and y coordinates) based on the face equation. Then, this distance is saved into the element with the same x,y coordinates of a 2-dimensional array. If in this element, a distance is already stored, the system compares the newly generated distance with the old one. Only the smaller one can be kept in this element (the Z-buffering principle). If the scan point is outside of all the 2-dimensional projections, the element is set to a prespecified background value. Finally, the range image is generated after each face of the object is scanned.

Appendix B

Proof of Lemma 1

The proof of Lemma 1 is given below.

The proof of the necessity of Lemma 1:

If the given transformation T is a rigid-body transformation ${}^{B}T_{A}$ for the whole set of corresponding point pairs, it will hold for any four corresponding point pairs among this set, and of course, for any four non-planar corresponding point pair (if they exist).

The proof of the sufficiency of Lemma 1:

Given a transformation T which can hold for four non-planar corresponding point pairs of the whole set $\{({}^{A}\vec{P}_i, {}^{B}\vec{Q}_i),\ i = 1\ \ to\ \ N\}$.

$$ {}^{B}\vec{Q}_j = T\,{}^{A}\vec{P}_j, \quad j = 1\ \ to\ \ 4. \tag{B.1} $$

Let $\{ {}^A\vec{U}_1, {}^A\vec{U}_2, {}^A\vec{U}_3 \}$ and $\{ {}^B\vec{V}_1, {}^B\vec{V}_2, {}^B\vec{V}_3\}$ be two sets of base vectors respectively in frame A and frame B,

$$ {}^A\vec{U}_1 = {}^A\vec{P}_2 - {}^A\vec{P}_1, \quad {}^A\vec{U}_2 = {}^A\vec{P}_3 - {}^A\vec{P}_1, \quad {}^A\vec{U}_3 = {}^A\vec{P}_4 - {}^A\vec{P}_1,$$
$$ {}^B\vec{V}_1 = {}^B\vec{Q}_2 - {}^B\vec{Q}_1, \quad {}^B\vec{V}_2 = {}^B\vec{Q}_3 - {}^B\vec{Q}_1, \quad {}^B\vec{V}_3 = {}^B\vec{Q}_4 - {}^B\vec{Q}_1.$$

From B.1, it follows

$$ {}^B\vec{V}_k = T\, {}^A\vec{U}_k, \quad k = 1 \ \ to \ \ 3. \tag{B.2}$$

Note that vectors ${}^A\vec{U}_1$, ${}^A\vec{U}_2$, and ${}^A\vec{U}_3$ (${}^B\vec{V}_1$, ${}^B\vec{V}_2$, and ${}^B\vec{V}_3$) are non-coplanar.

What needs to be proved is that the transformation T holds for all the corresponding point pairs in the set,

$$ {}^B\vec{Q}_j = T\, {}^A\vec{P}_j, \quad j = 1 \ \ to \ \ N. \tag{B.3}$$

Any point ${}^A\vec{P}_j, j = 1$ *to* N in frame A (${}^B\vec{Q}_j, j = 1$ *to* N in frame B) can be uniquely denoted by the linear combination of ${}^A\vec{U}_1$, ${}^A\vec{U}_2$, and ${}^A\vec{U}_3$ (${}^B\vec{V}_1$, ${}^B\vec{V}_2$, and ${}^B\vec{V}_3$),

$$ {}^A\vec{P}_j = a_{1j}\, {}^A\vec{U}_1 + a_{2j}\, {}^A\vec{U}_2 + a_{3j}\, {}^A\vec{U}_3 + {}^A\vec{P}_1 \tag{B.4}$$

and

$$ {}^B\vec{Q}_j = b_{1j}\, {}^B\vec{V}_1 + b_{2j}\, {}^B\vec{V}_2 + b_{3j}\, {}^B\vec{V}_3 + {}^B\vec{Q}_1 \tag{B.5}$$

where a_{1j}, a_{2j} and a_{3j} (b_{1j}, b_{2j}, b_{3j}) are the projections of ${}^A\vec{P}_j$ (${}^B\vec{Q}_j$) to the base vectors ${}^A\vec{U}_1$, ${}^A\vec{U}_2$, and ${}^A\vec{U}_3$ (${}^B\vec{V}_1$, ${}^B\vec{V}_2$, and ${}^B\vec{V}_3$) respectively.

Based on the initial definition that all the corresponding point pairs, $\{({}^A\vec{P}_j, {}^B\vec{Q}_j), \ j = 1 \ \ to \ \ N\}$, are related by a rigid-body transformation ${}^BT_A$, i.e. magnitudes of corresponding vectors are the same and angles

between two pairs of corresponding vectors are the same. Also, if ${}^{A}\vec{P}_k$ corresponds ${}^{B}\vec{Q}_k$ and ${}^{A}\vec{P}_l$ corresponds ${}^{B}\vec{Q}_l$, the projection of ${}^{A}\vec{P}_k$ to ${}^{A}\vec{P}_l$ is equal to the projection of ${}^{B}\vec{Q}_k$ to ${}^{B}\vec{Q}_l$.

Then it follows

$$b_{1j} = a_{1j}, \quad b_{2j} = a_{2j}, \quad b_{3j} = a_{3j}, \quad j = 1 \ to \ N. \tag{B.6}$$

Hence, for any $j = 1 \ to \ N$, it follows

$$\begin{aligned}
{}^{B}\vec{Q}_j &= b_{1j}\,{}^{B}\vec{V}_1 + b_{2j}\,{}^{B}\vec{V}_2 + b_{3j}\,{}^{B}\vec{V}_3 + {}^{B}\vec{Q}_1 \\
&= b_{1j}T\,{}^{A}\vec{U}_1 + b_{2j}T\,{}^{A}\vec{U}_2 + b_{3j}T\,{}^{A}\vec{U}_3 + T\,{}^{A}\vec{P}_1 \\
&= a_{1j}T\,{}^{A}\vec{U}_1 + a_{2j}T\,{}^{A}\vec{U}_2 + a_{3j}T\,{}^{A}\vec{U}_3 + T\,{}^{A}\vec{P}_1 \\
&= T\,(\,a_{1j}\,{}^{A}\vec{U}_1 + a_{2j}\,{}^{A}\vec{U}_2 + a_{3j}\,{}^{A}\vec{U}_3 + {}^{A}\vec{P}_1) \\
&= T\,{}^{A}\vec{P}_j
\end{aligned} \tag{B.7}$$

Therefor, the sufficiency of Lemma 1 has been proved.

REFERENCES

[1] J. Angeles. Automatic computation of the screw parameters of rigid-body motions. *Journal of Dynamic Systems, Measurement, and Control*, 108(3):32–38, March 1986.

[2] D.H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):714–725, 1981.

[3] S.T. Barnard and M.A. Fischler. Computational stereo. *ACM Comput. Surv.*, 14:553–572, 1982.

[4] D.I. Barnea and H.E. Silverman. A class of algorithms for fast digital image registration. *IEEE Trans. on Computers*, 21:179–186, 1972.

[5] H.G. Baumgart. Winged-edge polyhedron representation. Technical Report 320, Computer Science Department, Stanford University, Palo Alto, CA, 1972.

[6] H.G. Baumgart. *Geometric Modeling for Computer Vision.* PhD thesis, Computer Science Dept., Stanford Univ., Stanford, Calif., 1974.

[7] P.J. Besl and R.C. Jain. Three dimensional object recognition. *ACM Comput. Surv.*, 17:75–145, 1985.

[8] H. Chiyokura. *Solid Modeling with Designbase.* Addison-Wesley, 1988.

[9] C.I. Connoly. The determination of next best views. In *IEEE International Conference on Robotics and Automation*, pages 432–435, 1985.

[10] J.J. Craig. *Introduction to Robotics.* Addison Wesley, 1989.

[11] R.O. Duda. *Pattern Classification and Scene Analysis.* Wiley-Interscience, 1973.

[12] O.D. Faugeras and M. Hebert. A 3-D recognition and positioning algorithm using geometrical matching between primitive surfaces. In *The International Joint Conference on Artificial Intelligence*, pages 996–1002, 1983.

[13] F.P. Ferrie. *Reconstructing and Interpreting the 3D Shape of Moving Objects.* PhD thesis, McGill University, Montreal, Quebec, Canada, 1986.

[14] J.D. Foley, A.V. Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics*, pages 533–546. Addison-Wesley, second edition, 1991.

[15] P.C. Gaston and T. Lozano-Perez. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):257–266, 1984.

[16] R.C. Gonzalez and P. Wintz. *Digital Image Processing.* Addison-Wesley, 1977.

[17] W.E.L. Grimson. Computational experiments with a feature based stereo algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(1):17–34, 1985.

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

[19] W.E.L. Grimson and T. Lozano-Perez. Localizing overlapping parts by searching the interpretation tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.

[20] K.K. Gupta. Fast collision avoidance for manipulator arms: A sequential search strategy. *IEEE Transactions on Robotics and Automation*, 6(5), June, 1990.

[21] K.K. Gupta and Z. Guo. Toward practical motion planners: Experiments with a sequential search strategy. In *'91 ICAR Fifth Inter. Conf. on Advanced Robotics. Pisa, Italy*, pages 1006–1011, 1991.

[22] K.K. Gupta and X.M. Zhu. Extracting polyhedral models from a range image. In *Vision Interface '92, Vancouver, BC, Canada*, pages 37–42, 1992.

[23] E.L. Hall. *Computer Image Processing and Recognition.* Academic Press, 1979.

[24] R.M. Haralick. Solving camera parameters from the perspective projection of a parameterized cureve. *Pattern Recognition*, 17(6):637–645, 1984.

[25] R.M. Haralick and L.G. Shapiro. The consistent labeling problem: Part I. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:173–184, 1979.

[26] T.C. Henderson. Efficient 3*d* object representations for industrial vision systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:609–617, 1983.

[27] M. Herman. Matching three-dimensional symbolic descriptions obtained from multiple views of a scene. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 585–590, 1984.

[28] M. Herman. Generating detailed scene descriptions. In *IEEE International Conference on Robotics and Automation*, pages 426–431, 1985.

[29] M. Herman. Incremental reconstruction of 3D scenes from multiple, complex images. *Artificial Intelligence*, 30:289–341, 1986.

[30] M. Herman, T. Kanade, and S. Kuroe. Incremental acquisition of a three-dimensional scene model from images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):331–340, 1984.

[31] T.-H. Hong and M.O. Shneier. Describing a robot's workspace using a sequence of views from a moving camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(6):721–726, 1985.

[32] R.A. Hummel and S.W. Zucker. On the foundations of relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(3):267–287, 1983.

[33] K. Ikeuchi and B.K.P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, 17:141–184, 1981.

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

[35] R. Jain. Dynamic scene analysis. in progress in pattern recognition. In A. Rosenfeld and L. Kanal, editors, *Progress in Pattern Recognition.* North-Holland, Amsterdm, 1983.

[36] R.C. Jain and A.K. Jain. *Analysis and Interpretation of Range Images.* Springer-Verlag, 1990.

[37] R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):122–139, 1983.

[38] H.-S. Kim. *Active Multiple View Object Recognition.* PhD thesis, College of Engineering, The Univ. of Michigan, Ann Arbor, Michigan, 1988.

[39] H.-S. Kim, R.C. Jain, and R.A. Volz. Object recognition using multiple views. In *IEEE International Conference on Robotics and Automation*, pages 28–33, 1985.

[40] M.J. Magee and J.K. Aggarwal. Using multisensory images to derive the structure of three-dimensional objects — a review. *Computer Vision, Graphics and Image Processing*, 32:145–157, 1985.

[41] W.N. Martin and J.K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–173, 1983.

[42] M. Oshima and Y. Shirai. Object recognition using three-dimensional information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(4):353–361, 1983.

[43] A. Pentland, T. Darrell, M. Turk, and W. Huang. A simple, real-time range camera. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 256–261, 1989.

[44] S.B. Pollard, T.P. Pridmore, J.Porrill, and J.E.W. Mayhew J.P.Frisby. Geometrical modeling from multiple stereo views. *The International Journal of Robotics Research*, 8(4):3–32, 1989.

[45] A.A.G. Requicha. Representations for rigid solids: Theory, methods and systems. *Computing Surveys*, 12(4):437–464, 1980.

[46] M. Rioux, F. Blais, J.A. Beraldin, and P. Boulanger. Range imaging sensors development at NRC laboratories. In *Workshop on Interpretation of 3D Scenes*, pages 154–160, 1989.

[47] A. Rosenfeld. *Digital Picture Processing.* Academic Press, 1981.

[48] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6(6):420–433, 1976.

[49] J. Sklansky. On the Hough technique for curve detection. *IEEE Trans. on Computers*, 27:923–926, 1978.

[50] J.R. Stenstrom and C.I.Connolly. Building wire frames from multiple range views. In *IEEE International Conference on Robotics and Automation*, pages 615–620, 1986.

[51] S. Tanade and A. Rosenfeld. Point pattern matching by relaxation. *Pattern Recognition*, 12:269–275, 1980.

[52] F. Tomita and T. Kanade. A 3D vision system: Generating and matching shape descriptions in range images. In *The International Conference on Artificial Intelligence Application*, pages 186–191, 1984.

[53] R.Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 364–371, 1986.

[54] R.Y. Tsai. A versatile camera calibration technique for high-accuracy using off-the-shelf TV cameras and lenses. *The IEEE Journal of Robotics and Research*, 3(4):323–344, August 1987.

[55] B.C. Vemuri and J.K. Aggarwal. 3-D model construction from multiple views using tangential and intensity data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 435–437, 1986.

[56] E.L. Walker and M. Herman. Geometric reasoning for constructing 3D scene descriptions from images. *Artificial Intelligence*, 37:275–290, 1988.

[57] H. Wechsler. *Computational Vision.* Academic Press, 1990.

[58] S. Xie. *Incremental Construction of 3-D Models of a Scene from Sequentially Planned Views.* PhD thesis, School of Computing Scinece, Simon Fraser University, Burnaby, B.C. V5A 1S6, 1987.

[59] X.M. Zhu. Extracting polyhedral models from a range image - a hybrid approach. Master's thesis, School of Engineering Science, Simon Fraser University, Burnaby, B.C. Canada, 1991.