CAD-BASED COMPUTER VISION: MODELING AND RECOGNITION STRATEGIES

Dissertation for the Degree of Ph. D.

MICHIGAN STATE UNIVERSITY

PATRICK JOSEPH FLYNN

1990

CAD-BASED COMPUTER VISION:

MODELING AND RECOGNITION STRATEGIES

By

Patrick Joseph Flynn

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

1990

Abstract

CAD-Based Computer Vision: Modeling and Recognition Strategies

By

Patrick Joseph Flynn

In this thesis we develop a computer vision system for recognizing and locating three-dimensional objects in an industrial setting, where the objects have been designed on a commercial solid modeler (CAD system). Such a vision system can aid in automating labor-intensive manufacturing tasks such as inspection, classification, and packaging for shipment. The problem to be attacked is stated as follows: given a depth map containing one or more 3D objects which are piecewise-planar, piecewise-quadric, or piecewise-surface-of-revolution, identify and locate those objects which are instances of predefined 3D models stored in an object database.

Of particular importance to an object recognition system is the representation of 3D objects. The representations produced by present-day CAD systems are not organized efficiently for object recognition, and some salient features are not stored explicitly. One major contribution of this work is the development of a novel approach to object representation. We link a 3D boundary representation with powerful viewpoint-dependent features to produce augmented relational graph representations of objects.

One of the popular computational models for object recognition is interpretation-tree search. We have implemented a search-based recognition module which iteratively proposes identity and pose hypotheses for combinations of scene surfaces, and then discards hypotheses not in agreement with the scene content. Several hypothesis verification steps are used to prune the search space being explored; these steps involve comparison of the scene entities in the hypothesis and model features, and tests are based on intuitive geometric constraints such as proximity, visibility, and relative orientation.

This object recognition system is evaluated with both synthetically-generated depth data from CAD models of twenty objects and depth data acquired from a triangulation-based 3D range finder. An input image is segmented into connected surface patches using a region-based local-feature clustering technique, followed by patch merging at domain-independent and domain-dependent levels. Surface patches are classified using regression and surface curvature estimates. Classified patches, their parameters, edge positions, and edge lengths serve as input to the identification/localization module. This system is implemented on a graphics workstation in the C and Common LISP languages, and tested on a variety of 3D objects.

To my parents,
and Laurie June Starr

Acknowledgements

The research described in this thesis would not have been possible were it not for the continuous support of my thesis advisor, Professor Anil Jain. Through my years in graduate school, he has constantly been supportive during the rough times, and cracked the whip over me during my occasional lazy periods. I deeply appreciate our hundreds of hours of discussions, and value him as a close friend. I also gratefully acknowledge the assistance of Professor George Stockman, another of my committee members, with whom I have spent many fruitful hours in conversation. I also appreciate the assistance and interest of Professor Lionel Ni, Professor Richard Hill, and Professor Erik Goodman, all members of my thesis committee. I thank Dr. Goodman for making the facilities of the Case Center for Computer-Aided Engineering and Manufacturing available to me during my thesis research. I spent three enjoyable months during the summer of 1987 as an intern in the Automation Sciences Laboratory of the Northrop Research and Technology Center in Palos Verdes, California. Dr. H. R. Keshavan, the laboratory director and the sixth member of my thesis committee, has been generous with his time and resources during and after my internship at Northrop. I value his many comments and practical advice highly. Professors Richard Dubes and Mihran Tüceryan have also given me many useful ideas over the past few years. I appreciate their guidance. I also value the counsel of Dr. Torfinn Taxt of the Norwegian Computing Centre and Professor Guna Seetharaman of the University of Southwest Louisiana, both of whom spent several months at MSU as visiting scholars. Dr. Paul Besl of General Motors Research Laboratories and Professor W. Eric L. Grimson of the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology gave advice at crucial points during my research.

The Pattern Recognition and Image Processing Laboratory has been an excellent environment in which to carry out graduate research and yet preserve one's sanity during the high-pressure periods. In no small part, this is due to the concern and encouragement of others, particularly my graduate student comrades. Many thanks to

all current and former graduate students for their help, especially Dr. Rick Hoffman, Dr. Gongzhu Hu, Dr. Sei-Wang Chen, Dr. Bruce McMillin, Henry Davies, Ray Klein, Yhwyng Lee Hoffman, Joe Miller, Farshid Farrokhnia, Deborah Trytten, Narayan S. Raja, Satish Nadabar, Tim Newman, Arun Nanda, and Sally Howden. Special thanks go to David Marks, our laboratory manager, for his assistance and the occasional trip out for pizza.

Thanks to Drs. Roland Zapp and Marvin Siegel of the Electrical Engineering Department, and Gregory Hoshal and Bill Byrne, for helping me get my start in research as an undergraduate.

This research has been supported by the National Science Foundation under grants IRI-8600371 and CDA-8806599, by the Department of Computer Science, by the GTE Graduate Fellowship Program, and by the Northrop Research and Technology Center. I also appreciate the financial support of Dr. Reddiar S. Anbalagan of Innovision Corporation of Madison, Wisconsin.

Finally, I acknowledge with love the support of my parents, my family, my wife's family, and especially my wife Laurie, who kept me motivated.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis describes an object recognition system employing geometric models of 3D objects constructed on a computer-aided design (CAD) system and range (depth) imagery; given these two inputs, it identifies instances of the object models present in the scene and estimates their location and orientation. Such a system can be used successfully in automated manufacturing environments to perform inspection, assembly, sorting, bin-picking, and classification tasks with the aid of a robotic manipulator. In this chapter, we motivate the object recognition and localization problem, the role of CAD models, and the use of range imagery. An outline of the remainder of the thesis concludes the chapter.

1.1 Manufacturing and Computer Vision

Modern manufacturing enterprises are becoming increasingly automated. Computer technology is present in one form or another in almost every manufacturing plant, regardless of the products produced, the quantity produced, or the intended consumer. The primary goals of automation are

- improved product quality,
- reduced manufacturing cost, and
- reduced lead time.

Benefits derived from automation can include increased production rates and improved consistency. The primary requirement for successful automation in manufacturing is the *integration* of information between design, fabrication, testing, and inspection. Integration allows design integrity to be maintained throughout this sequence. The most important information involved in this integration is *geometric* (specifically, solid models of the objects to be fabricated).

One of the most successful applications of computers in manufacturing has been the use of CAD software to design mechanical parts. Over the past decade, many draftsmen have been replaced by engineers with desktop workstations; product specifications are converted to mathematical descriptions of the parts to be manufactured, and these descriptions are stored 'on-line', without resorting to paper copies. In situations where the mechanical parts can undergo stresses, finite-element techniques and simulation studies can validate a design against specifications, often eliminating or reducing the fabrication and destructive testing of prototypes. Finally, when a design is found to meet all manufacturing requirements, the geometric model of the object residing in the computer can be translated into process plans and programs for computer-driven numerical control machines which fabricate the part from stock materials.

The scenario described above is being carried out at many manufacturing facilities. However, the usefulness of geometric models in automated manufacturing certainly does not end there. The last decade has seen much progress in computer vision techniques and their applications in industrial environments. Vision systems are used in the factory for a number of tasks [68]:

- *Metrology*: verification of dimensioning tolerances;
- *Surface Inspection*: examination for proper finish and absence of other surface flaws;
- *Integrity and Placement Verification*: checking for presence and proper positioning of subparts;
- *Model Acquisition*: construction of an object description from an image or a series of images;
- *Navigation*: provide information about the environment to a teleoperated or autonomous vehicle;
- *Feature Localization*: identify and locate a particular feature (such as a hole or corner) on an object, without finding the overall position and orientation of the object; teleoperated or autonomous vehicle;
- *Recognition*: identification of a part as one of several that might be seen by the sensor; and

- *Object localization*: estimation of the position and orientation of the part in the sensor's field of view.

In many of these vision applications, geometric information about the manufactured part(s) is useful or even essential. For example, to verify that a given dimension of the part being viewed is within a specified tolerance of the design value, one needs to know the design and tolerance values. Clearly, the geometric model produced by the part designer can provide much, if not all of the object information required by an industrial vision system. Paradoxically, though, the integration of CAD information into industrial vision systems has only recently been attacked in its own right [11]. As computer vision research matures and industrial applications of vision assume more importance in manufacturing, this issue of integration is becoming an area of serious research.

Part of the contribution of this thesis is a method of integrating geometric models designed on a mechanical CAD system into a model-based 3D object recognition and localization system.

Of the several industrial vision problems outlined above, the latter two (recognition and object localization) have received much attention from computer vision researchers but they remain difficult and challenging problems. Computer vision research has recognized that two fundamental problems underlie the recognition and localization task: *representation* of 3D objects, and *matching* of stored representations to the sensed image of the scene. The representation problem has been studied by researchers in computer graphics, CAD, and vision; computer vision systems can benefit from the prior work done by others. However, the matching problem has only recently been addressed in sufficient generality to be of interest in automated manufacturing. In the sequel, we will refer to the tasks of recognition and localization jointly as the *recognition problem*; in Chapter 5, recognition and localization are attacked as a single unified task. The second major contribution of this thesis is the implementation of a 3D object recognition system which takes as its input a *range image* (depth map) of a scene with one or more objects, possibly in self-occluding or mutually-occluding positions, and produces as its output a listing of (known) objects in the scene along with their locations and orientations.

1.2 Three-Dimensional Sensing

The discussion above has motivated the usefulness of computer vision in the manufacturing environment, and suggested a multifaceted role for object geometry in applications involving sensing of the manufactured parts. However, the actual extraction of surface geometry is a difficult problem. Many prior vision systems have employed intensity sensors such as CCD (charge-coupled device) cameras to obtain images. The resulting images contain information about surface finish, ambient and active lighting, and other extraneous information as well as the desired information about the scene geometry. The scene geometry is present in *implicit* form, and the difficult task is to extract it explicitly. Computer vision researchers continue to attack the problem of extracting depth information from intensity images using a variety of 'shape-from-X' approaches such as shape from shading [59], shape from texture [121], and shape from occluding contours [115][1].

Recent years have seen the development of a variety of techniques for sensing depth (range) information directly. A survey of such techniques appears in Chapter 3; here, we broadly classify range sensors into one of four types [7]:

- Radar sensors: time-of-flight, AM, and FM detectors;
- Structured light sensors: triangulation-based techniques;
- Stereo: methods using multiple cameras or light sources; and
- Miscellaneous methods: Moire fringe analysis, holographic interferometry, and depth-from-focus.

Structured light sensors obtain depth information by projecting a line, grid, or other pattern on the scene, which is sensed by an intensity sensor at a known location. Knowledge of the pattern's geometry and the camera's position allows the depth to be derived from the distorted pattern seen by the intensity sensor. In this dissertation, images of 3D parts are obtained from a structured light scanner [113] in our laboratory; images from other sensors (*e.g.* radar range finders) could also be used.

[1] Many of these approaches are supported by biological evidence; human beings do quite well at recovering shape from intensity imagery. Some animals use different sensing strategies to find the 'shape' of their environment (*e.g.*, bats use a radar-like sense).

1.3 3D Object Recognition

A 3D object recognition system employs data from two sources to interpret a scene:

- A *model database*, containing representations of the objects to be recognized, and

- A *scene representation*, which summarizes the scene content extracted from the sensed image(s).

The process of identifying a model object which is present in the scene is referred to as *matching*, *interpretation*, *recognition*, or *identification*. Many systems, including the one developed in this dissertation, also perform *localization*, the estimation of a rigid rotation and translation required to bring the coordinate system of the identified model into correspondence with the scene coordinate system.

We will defer a survey of existing object recognition systems to Chapter 5 and limit our discussion to a taxonomy of recognition techniques. Figure 1.1 illustrates two design dichotomies in object recognition systems. Designers must choose a *representation strategy* for their object models; the primary distinction deals with dimensionality. Are models represented internally as a series of 2D views (*e.g.* silhouettes), or are model curves and surfaces expressed geometrically in a common 3D coordinate system? Additionally, the nature of the sensed image affects the recognition strategy. The two major types of imagery in use today are intensity (or 2D) imagery, and range (or 3D) imagery[2]. The choice of object representation strategy and image type results in the following four types of object recognition systems:

- *2D-2D* (2D image, 2D model) matching strategies are often practical in constrained recognition environments.

- *2D-3D* (2D image, 3D model) strategies have been popular topics of research because they combine the descriptive power of 3D models with inexpensive and fast 2D sensing strategies.

- *3D-2D* (3D image, 2D model) techniques have not been studied in any depth. Matching a 3D image against a 2D model is wasteful of the image data. For

[2] Not all researchers would agree with this classification. Some point to work in shape-from-shading as an argument that intensity imagery is 'more than 2D'; some view range imagery as $2\frac{1}{2}$D because data from the 'underside' of the object is not obtained, and reserve the 3D label for stacks of density maps (*e.g.* MRI images) from which data on all surfaces of an object can be obtained.

		Type of image	
		2D	3D
Type of Model	2D	2D-2D	3D-2D
	3D	2D-3D	3D-3D

Figure 1.1: Design dichotomies in object recognition systems.

example, we would be discarding almost all of the range data in a system which matches 2D silhouette models to 3D images.

- *3D-3D* (3D image, 3D model) methods have received much attention recently; our system falls in this category. An additional point of departure in this area is the source of object models. One contribution of our work is the use of CAD models in the recognition system; many prior systems use hand-designed models, incomplete or coarse models, or models learned from multiple images.

3D representations are studied in detail in Chapter 2. We should note, however, that the issue of *power* in object representation has historically been a barrier to the use of 3D vision systems in industrial applications. Most of the early systems employing 3D models were limited to either polyhedral or generalized cylinder representations for objects. Neither approach captures enough variation in 3D geometry to adequately and unambiguously represent a large class of manufactured parts. A versatile vision system using 3D models should allow objects constructed using constructive solid geometry (primitive shapes and Boolean combinations), sweeps, profiling, and extrusion methods to be represented exactly. Our system is an improvement over polyhedral representations, but does not attempt to represent objects with complex sculptured surfaces.

1.4 Synthesis: CAD-Based 3D Vision

In previous sections we have identified problems in the manufacturing domain that can be addressed using computer vision techniques. We have also highlighted the central role of object geometry in these applications. Range sensors can provide a computer with a dense sampling of 3D points on an object's surface.

This thesis describes a vision system which recognizes and locates three-dimensional objects in an image of a scene obtained by a range sensor. Models of the objects to be recognized are constructed using a commercial CAD system (GEOMOD [110]), and stored in a database. Each object is described in the database as a list of *surfaces* (planes, cylinders, and spheres) and *bounding curves* (lines, circular arcs, and parametric curves). Higher-level view-dependent and view-independent geometric features are inferred from the geometric model produced by the CAD system, producing a *vision model* used during the matching stage. A block diagram of the system appears in Figure 1.2.

One important contribution of this thesis is its approach to creation of the object models against which the scene description is matched. Rather than creating models by hand, or building models by merging information from several views of the object, our models are constructed *automatically* from a CAD database. A number of geometric inferencing steps are performed on the CAD descriptions in order to extract object features that are useful in matching. The end result of the inferencing system is a relational graph model that contains a large amount of *explicit* geometric information about the objects. This information is necessary in building efficient object recognition strategies.[3]

The scene information to be processed is in the form of a range image, which contains explicit measurements of distance between the sensor and objects in the scene. This image is segmented to extract homogeneous (smooth) surface regions. From regions, geometric features (principal surface curvatures and directions), which characterize the 'local' shape of the surface at each point, are calculated. These

[3] Automatic construction of vision models from CAD models is useful in environments where new objects are being designed. In some environments, however, one might wish to create vision models for objects which already exist. If so, the construction of models from multiple views [91, 116] might be preferable to the effort of measuring the object, entering the measurements into the CAD system, and building a vision model using our approach. In a sense, model-building from multiple views and model-building from CAD models are complementary operations, and each has its place in 3D vision systems for industrial object recognition.

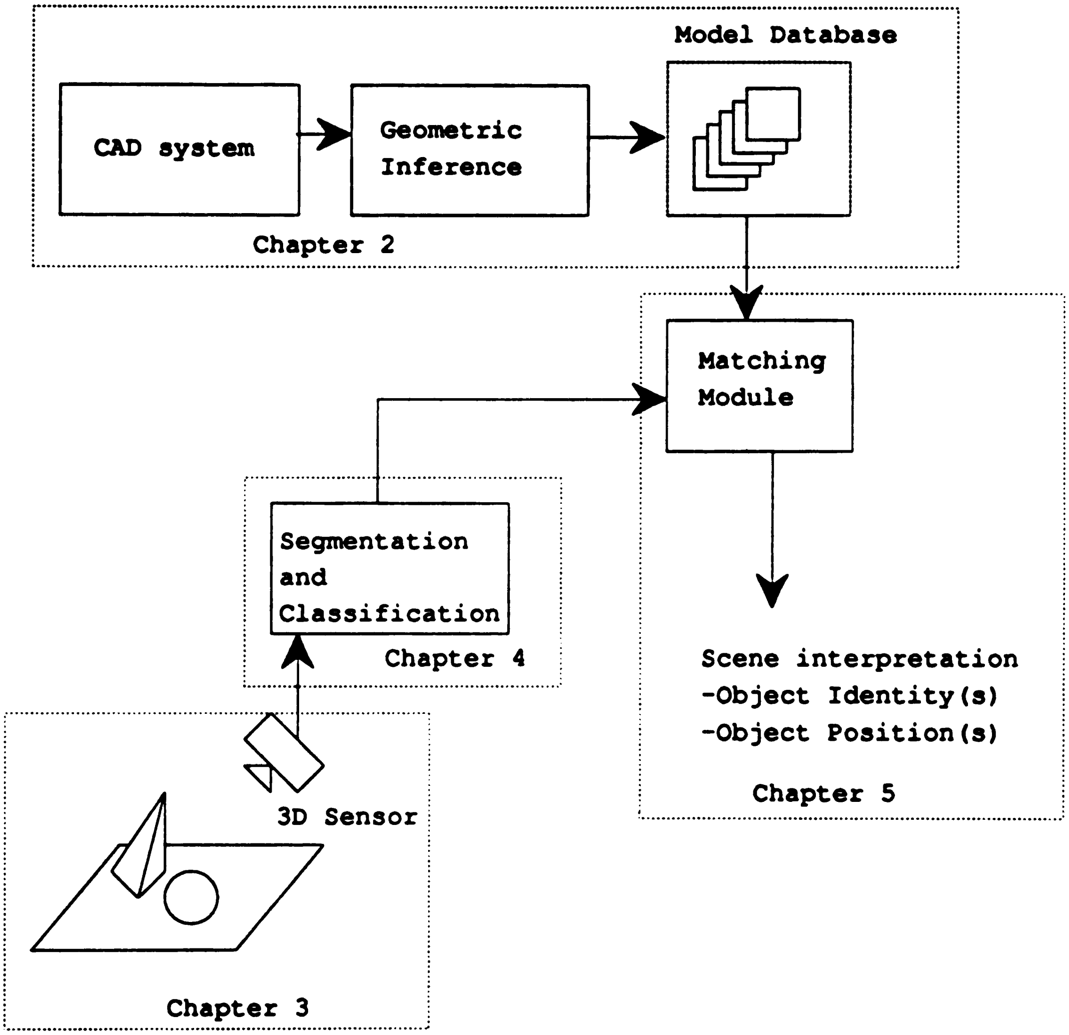

Figure 1.2: Block diagram of the CAD-based computer vision system

features are used to classify surfaces (*e.g.* planar, spherical) and estimate their parameters (*e.g.* the radius and orientation of a cylindrical patch). The extraction of smooth surfaces and their parameters from the image is done in the *segmentation* and *classification* module in Figure 1.2.

The output of the segmentation step is a list of classified surfaces in the image. The recognition module (named BONSAI) takes this information and attempts to match it against objects in the model database. The strategy used for recognition is the interpretation-tree search paradigm. BONSAI makes tentative hypotheses of identity for scene surfaces and model surfaces, attempts to extend these hypotheses to include additional scene and model surfaces, and rejects those hypotheses which are inconsistent with respect to the model under consideration. A number of constraints are used to reject hypotheses so that the search tree can be pruned. The second major contribution of this work is a detailed examination of the constraints on interpretation provided by object models and scene geometry. These constraints allow inconsistent hypotheses of object identity and pose to be rejected early in the recognition process, allowing a larger model database to be examined efficiently.

1.5 Outline of the Dissertation

The remainder of this dissertation is organized as follows. In Chapter 2, we survey three-dimensional object representation, describe the integration of geometric models constructed on a CAD system into our vision system (with particular emphasis on geometric inference), and describe our model database. Chapter 3 outlines the principles of 3D range sensing, describes the Technical Arts 100X sensor used to obtain our range images, and contains example images of the objects used in testing of the BONSAI system. Chapter 4 describes the segmentation and classification procedures used to obtain a symbolic scene description from an input depth map. Chapter 5 describes our search-based recognition system. Experiments using a variety of object models and sensed scenes appear in Chapter 6. Finally, Chapter 7 summarizes our results and recommends future directions for research in 3D object recognition.

Chapter 2

Representation: Formalism, CAD Models, and Vision Models

2.1 Introduction

In Chapter 1, we motivated the commonality of geometric knowledge between design, fabrication, and visual tasks such as 3D object recognition and localization. In essence, we wish to have one logical (not necessarily physical) geometric knowledge base for the entire manufacturing cycle, with representational modifications as necessary, driven by the requirements of the processes involved. While a complete discussion of such representations and modifications is beyond the scope of this dissertation, it is important to discuss in some detail popular geometric representations and describe those used in this thesis.

In this chapter, an outline of the mathematical properties of solid models is followed by descriptions of the popular methods for representing three-dimensional objects in commercial CAD systems. Then, we focus on the geometric modeling system (GEOMOD) [110] used in this research to create CAD models. A survey of representations used in computer vision research follows. The remainder of the chapter describes our CAD-to-vision model inferencing system, including examples of the model construction process for a few typical 3D objects used in this thesis.

2.2 Formal Properties of Geometric Representations

In this section, we discuss the mathematical formalism of 3D solid representations. Excellent surveys of solid modeling appear in Requicha [95], Requicha and Voelcker [93, 94], and Brown [23].

A *representation method* [95] $\mathcal{M}$ is a relation involving sets containing solid objects

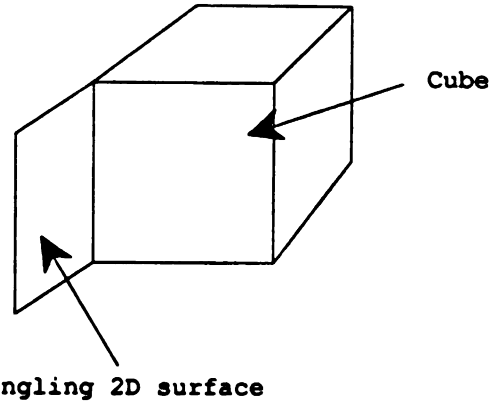

Figure 2.1: A 'nonsense' object.

and representations of solid objects. A member $m_i \in \mathcal{M}$ is an ordered pair (o_i, s_i), where $o_i \in O$ is an object (or a collection of objects) drawn from a set of objects and $s_i \in S$ is a representation (a *model*) of o_i drawn from the set of syntactically-correct representations (Requicha observes that S is a language generated by a grammar using a finite alphabet). Let $D \subset O$ and $R \subset S$ be the domain and range of $\mathcal{M}$. Elements $o \in (O - D)$ are not representable by $\mathcal{M}$, and elements $s \in (S - R)$ are syntactically-correct representations with no corresponding physical object (they are often called *invalid* representations).

Once the concept of representation has been expressed mathematically, we can ask some standard questions about the relation $\mathcal{M}$:

- How large is the set D? This allows us to judge the *power* of the representation scheme. A representation $\mathcal{M}_1$ admitting only polyhedra is less powerful than a scheme $\mathcal{M}_2$ admitting natural quadrics, because the domain of $\mathcal{M}_1$ is a proper subset of the domain of $\mathcal{M}_2$.

- How large is R? If $R = S$, the representation scheme does not admit 'nonsense' objects (*e.g.* an object with a 'dangling' surface; see Figure 2.1). While it is difficult to imagine a human designer intentionally creating such 'nonsense' objects, a procedure for automatically detecting and flagging such models would be useful in situations where they are generated by a computer.

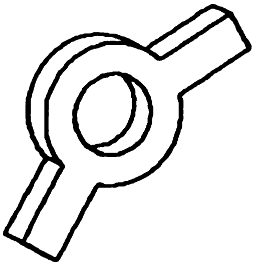

Figure 2.2: A 3D object. Various representations for this part appear in the text.

- Are representations *complete* (unambiguous)? A *representation s* is unambiguous if

  $$(o_1, s) \in \mathcal{M} \wedge (o_2, s) \in \mathcal{M} \Rightarrow o_1 = o_2.$$

  The *method* $\mathcal{M}$ is unambiguous (formally, *left unique*) if

  $$(o_1, s) \in \mathcal{M} \wedge (o_2, s) \in \mathcal{M} \Rightarrow o_1 = o_2 \; \forall o_1, o_2 \in D \; \forall s \in R.$$

  If $\mathcal{M}$ is ambiguous, representations do not contain enough information to reconstruct the 3D object. In 3D vision, complete models are desirable for a pragmatic reason: they can allow synthetic images (intensity and range) to be produced from the model.

- Are representations *unique*? Unique (formally, *right unique*) schemes are functions from O to S; they satisfy

  $$(o, s_1) \in \mathcal{M} \wedge (o, s_2) \in \mathcal{M} \Rightarrow s_1 = s_2 \; \forall o \in D \; \forall s_1, s_2 \in R.$$

  If not, more than one representation in R corresponds to a single physical object. In commercial CAD systems, this situation can easily arise, since different design methodologies can produce the same object.

The 'ideal' 3D representation method produces only valid representations, is unique and unambiguous (and hence one-to-one), and also has a rich domain of representable objects. Most 3D representation methods fail one of these criteria; this partially explains the variety of approaches used by researchers to represent 3D solids.

To some degree, the particular representation strategy used in model-building can be 'hidden' from the end user of the modeling software. Most commercial solid modelers allow the user to build objects using constructive solid geometry (CSG) techniques (primitives and Boolean operations; see below). However, the internal representation of the solid so constructed need not be a CSG representation. Indeed, some solid modelers maintain *multiple representations* of objects under construction, in order to offer the user flexibility of design, speed of display, and representational power. For example, the GEOMOD software used in this research contains three distinct representations of each object.

- A boundary representation (B-rep), where curves and surfaces are nonuniform rational B-splines.
- A 'CSG-like' tree describing the design history in terms of primitives and operations.
- A polyhedral approximation to the object's boundary (which is useful for displaying the object during the design cycle).

Hoffmann briefly discusses the issues of abstraction in solid modeling, and the usefulness of multiple representations [58, pp. 6-14]. Of course, the use of multiple internal representations introduces a coherency problem. How does the modeler ensure that *all* of the internal representations describe identical objects? Discussions of this problem are beyond the scope of this dissertation.

2.3 Representation Schemes

This section describes the dominant 3D object representation methods in use today: CSG, B-rep, volumetric techniques, sweep representations, and view-centered approaches. Most of these representation schemes has been used by vision researchers for the representation of 3D shapes. Besl and Jain [9] have examined 3D modeling from the vision perspective. Samet [98] gives detailed explorations of data structures suitable for internal representations of 2D and 3D solids. The model-building system developed in this thesis combines the representational power of B-reps with view-centered information to,produce a relational object model; to our knowledge, such a combination of approaches is unique.

For reasons of clarity, a common mathematical notation for models produced by different representation schemes is adopted. All representations will be denoted by

$\mathcal{R}$ with a subscript specifying the type; *e.g.* $\mathcal{R}_{CSG}$ will represent a CSG model of a given object. In addition, we will give an example of most representation schemes for the object shown in Figure 2.2.

2.3.1 Representations used in CAD Systems

Here, we discuss representation methods used in many CAD systems. Computer vision research employing these representations is noted in the descriptions.

Wire-frames

A wire-frame object representation is a graph

$$\mathcal{R}_{WF} = \{\mathcal{V}_{WF}, \mathcal{E}_{WF}\},$$

where the vertices are 3D points on the object surface (typically corners or other distinguished points), and the graph edges represent a physical edge on the object. These representations do not contain surface information and are therefore not complete, since the 'solid' portion of the wire frame is not defined in the graph structure. Few 3D vision systems have employed wire-frame representations.

Constructive Solid Geometry (CSG)

In CSG-based modeling, a finite set of primitive shapes (often simple shapes such as right circular cylinders, spheres, and boxes, but occasionally more general shapes or halfspaces) are combined using regularized Boolean operations to produce the desired solid. A CSG model is stored as a tree

$$\mathcal{R}_{CSG} = \{\mathcal{V}_{CSG}, \mathcal{E}_{CSG}\},$$

with vertex set $\mathcal{V}_{CSG}$ and edge set $\mathcal{E}_{CSG}$. Edges of the tree enforce precedence among the Boolean operations used to construct the object. If the vertex set is partitioned into subsets containing leaf and interior vertices,

$$\mathcal{V}_{CSG} = \mathcal{V}_{leaf} \bigcup \mathcal{V}_{interior},$$

then $\mathcal{V}_{leaf}$'s members are ordered pairs of the form

$$v_{leaf} = (type, parameters)$$

where *type* denotes the type of primitive (*i.e.* cylinder, cone, box), and *parameters* gives both the intrinsic parameters of the primitive (*e.g.* the cylinder's radius and length) and the location and orientation of the primitive in 3D. Each member of $\mathcal{V}_{interior}$ is one of the (regularized) Boolean operations UNION ($\cup*$), INTERSECTION ($\cap*$), and DIFFERENCE ($-*$).[1] A CSG tree for the object in Figure 2.2 appears in Figure 2.3. The intrinsic and location parameters of the primitives at the leaves are not shown. Lin and Chen [77] developed a 3D vision system which extracts CSG object representations from range data and matches them against a CSG model database.

Surface Models (B-reps)

A more powerful alternative to the CSG representation is the boundary representation. A B-rep can be viewed (conceptually) as a triple

$$\mathcal{R}_{Brep} = \{\mathcal{S}_{Brep}, \mathcal{I}_{Brep}, \mathcal{G}_{Brep}\},$$

with $\mathcal{S}_{Brep}$ the set of surfaces of the object, $\mathcal{I}_{Brep}$ a set of space curves describing the intersections between the surfaces in $\mathcal{S}_{Brep}$, and

$$\mathcal{G}_{Brep} = \{\mathcal{V}_{Brep}, \mathcal{E}_{Brep}\},$$

a graph describing the surface connectivity. Vertices in $\mathcal{V}_{Brep}$ represent the surfaces in $\mathcal{S}_{Brep}$, and edges express connectivity. Care must be taken to resolve any possible ambiguities in representation due to the sense (or polarity) of neighboring surfaces. Consider the two objects in Figure 2.4. The only difference between these two objects is the sense of the top cylindrical patch (convex on the left, and concave on the right). This difference could be expressed in the connectivity graphs $\mathcal{G}_{Brep}$ (although the difference in sense might also be expressed via different parametrizations of the cylinders or the cylinder-plane creases).

Figure 2.5 shows a portion of the B-rep model for the object in Figure 2.2 (the entire representation would be difficult to draw in a legible way on paper). Note the presence of individual surfaces and connectivity information (the small arcs).

Surfaces (*i.e.*, elements of $\mathcal{S}_{Brep}$) can be represented in a variety of ways [8]. Three of the most popular approaches are

[1] A *regularized* Boolean operation will produce a resultant object without dangling 2D or 1D artifacts. Regularized operations are conventionally denoted with a star following the logical operand (*e.g.* $\cup*$).

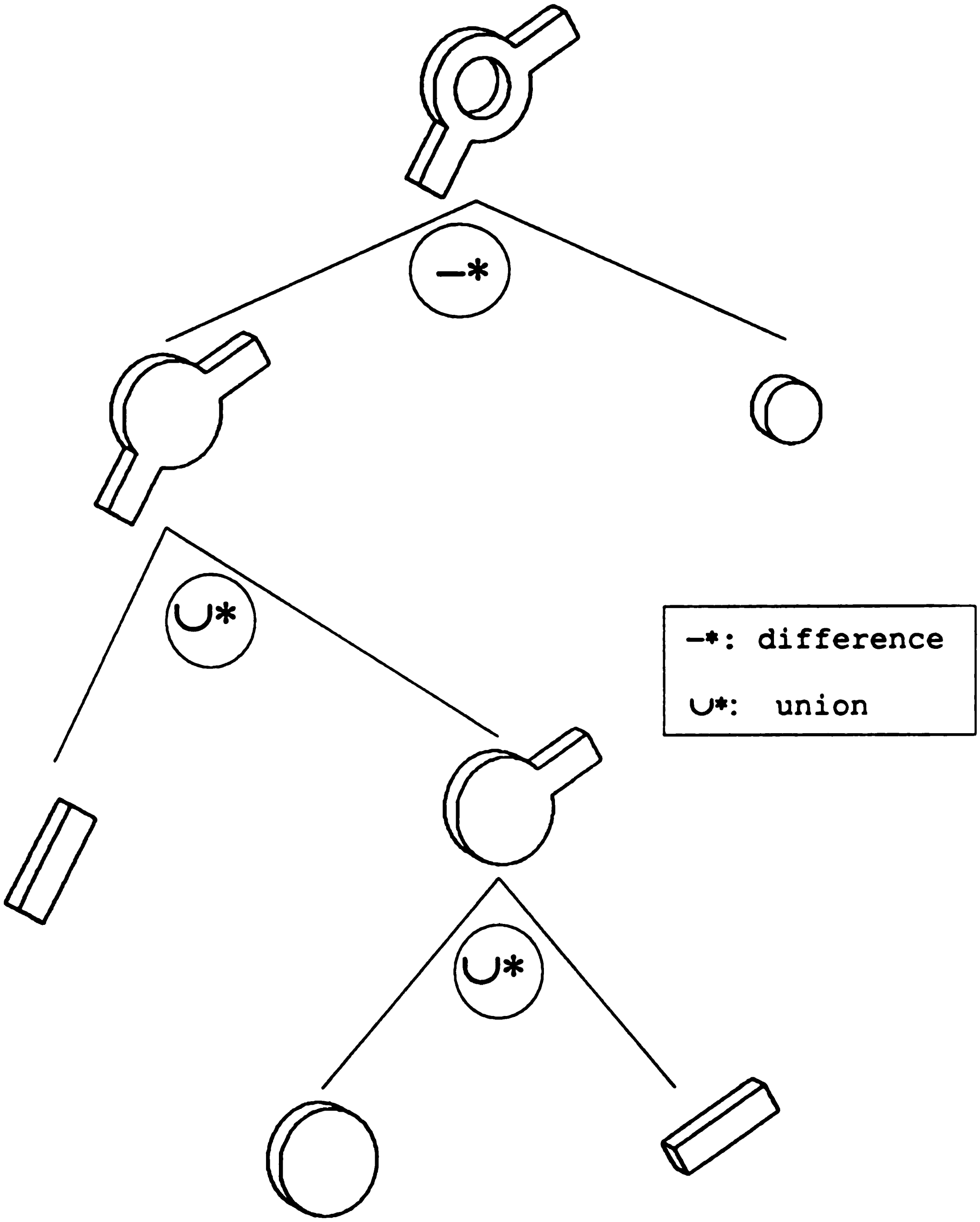

Figure 2.3: A CSG tree for the example object of Figure 2.2.

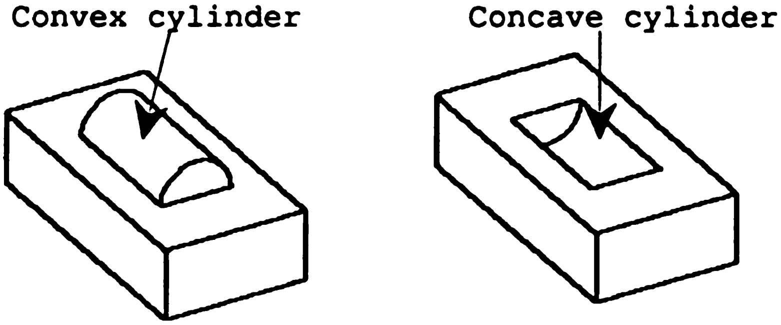

Figure 2.4: Possible ambiguities in boundary representations.

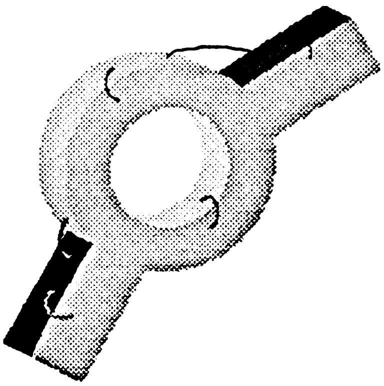

Figure 2.5: A portion of the B-rep for the example object.

- the *implicit* surface form

$$S = \{(x,y,z) : f(x,y,z) = 0\},$$

- the *general parametric* form

$$S = \{(x,y,z) : x = f_1(u,v), y = f_2(u,v), z = f_3(u,v), (u,v) \in D \subset \mathbf{R}^2\}, \text{and}$$

- the *graph* surface form

$$S = \{(x,y,z) : z = f(x,y), (x,y) \in D \subset \mathbf{R}^2\},$$

Clearly, the graph surface is just a special case of the general parametric surface; we highlight it because range data is usually viewed as samples from a graph surface (the range value z is a function of a point in the xy plane). The problem of fitting surfaces to range data can be tackled using regression techniques [39] when the graph surface representation is used.

Polyhedral models (in which the elements of $\mathcal{S}$ are planes and elements of $\mathcal{I}$ are lines) are the simplest boundary representations. Polyhedra are also very compact descriptions, requiring four scalars per surface. The primary drawback of polyhedral models is their representational power; objects with curved surfaces cannot be represented exactly, and polyhedral approximations to curved objects are not unique.

The next 'simplest' surface representation is the *quadric* surface. Implicit quadrics, defined by

$$S = \{(x,y,z) : a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z + a_{10} = 0\},$$

can represent many of the surface types encountered in manufacturing, such as spheres, cylinders, planes, and cones. If the implicit quadric is represented in general form, ten parameters are required per surface patch. However, Goldman [44] has noted that if the quadric *type* and its geometric parameters are stored instead, as few as five parameters are required. This 'geometric' representation for quadric primitives enjoys other advantages over the implicit equation when the problem of fitting quadrics to sensed data is considered. If one knows the surface type *a priori*, one can employ nonlinear optimization to get the best approximating quadric of that type, whereas traditional least-squares techniques pay no attention to surface type in fitting the surface. In [39] we explored the nonlinear fitting method as part of a surface classification strategy; one of the techniques developed there will be used in Chapter 4 for surface classification.

Most B-rep CAD systems use the general parametric form to represent surfaces; in particular, GEOMOD uses nonuniform rational B-spline surfaces of the form

$$S = \{(x,y,z) : x = f_1(u,v), y = f_2(u,v), z = f_3(u,v), \text{where}$$

$$[f_1\ f_2\ f_3] = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} B_{ik}(u)B_{jl}(v)h_{ij}\mathbf{P}_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} B_{ik}(u)B_{jl}(v)h_{ij}}. \tag{2.1}$$

Here, n and m are the number of 3D control points $\mathbf{P}_{ij} = [px_{ij}\, py_{ij}\, pz_{ij}]$ in the u and v parameter directions (for a total of nm points $\{\mathbf{P}_{ij}\}$), k and l are the orders of the B-spline curves along the u and v directions, and the $B_{ik}(u)$ and $B_{jl}(v)$ are the values of the ith B-spline basis polynomial of degree k at u (respectively, the jth degree-l basis polynomial at v), and the h_{ij} are control point weights associated with the $\mathbf{P}_{ij}$, specifying their relative contribution to the surface shape.

B-rep surface models have been slow to gain acceptance among vision researchers; two reasons for this can be cited. First, few of the true three-dimensional model-based recognition systems have required the functionality that can be expressed through the CAD methods. For example, recognition systems which work exclusively with polyhedral objects have no need to employ curved surface patches and bounding curves. Secondly, the popular surface representations in CAD (tensor-product splines) often convey little intuition about the object surface being modeled. In this respect natural surfaces like quadrics and planes have greater utility as modeling primitives. The difficulties of fitting more complicated surfaces to image data, *and employing the power of those descriptions for higher-level tasks* have also hindered their use by vision researchers.

Volumetric Models

One way to represent an object is to place it in some reference coordinate system and subdivide the volume it occupies into small 'primitive' volume elements (often called *voxels*). Most volumetric descriptions employ box-shaped primitives (cubes or parallelepipeds) as voxels. If only one size of box is used such descriptions can occupy a large amount of space, since a large number of boxes might be required to approximate a large object with a complicated boundary. This volumetric model can

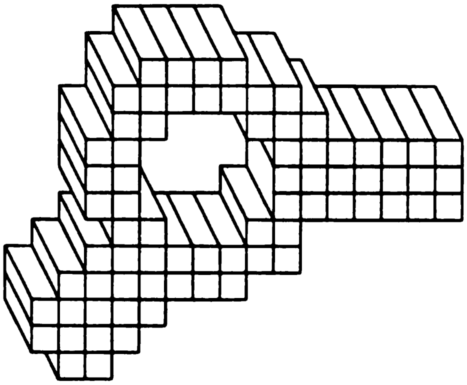

Figure 2.6: Volumetric approximation of the example object.

be represented internally as

$$\mathcal{R}_{voxel} = \{(l_x, l_y, l_z), \mathbf{p}_1, \mathbf{p}_2, \ldots \mathbf{p}_n\}, \text{where}$$

l_x, l_y, and l_z are the dimensions of the primitive box, and each $\mathbf{p}_i$ fixes the location of one of the boxes. Figure 2.6 shows a voxel representation of the example object. A more accurate representation could be obtained (at the cost of storage and processing time) by shrinking the box primitives. Some compression in size can be achieved by encoding the occupied volume in *octree* format [98]. Other representations based on volumetric primitives (such as spheres) have been proposed [88].

A fairly recent addition to the set of modeling primitives are the family of ***superquadric*** (or superellipsoidal) solids. The canonical superquadric's shape is controlled by two parameters ϵ_1 and ϵ_2; the expression for a point on a canonical superquadric at latitude η and longitude ω is given by

$$(x, y, z) = (\cos^{\epsilon_1} \eta \cdot \cos^{\epsilon_2} \omega, \cos^{\epsilon_1} \eta \cdot \sin^{\epsilon_2} \omega, \sin^{\epsilon_1} \eta),$$

and the entire closed solid is obtained by varying η through the domain $[0, \pi]$, and ω through the domain $[0, 2\pi]$. These volumetric primitives can then be deformed by ***bending***, ***twisting***, and ***tapering*** [4] to create a wide variety of curved primitives. Boolean combinations can be used to represent arbitrarily complicated shapes. Most 'interesting' 3D objects would be composed of a number of superquadric primitives placed in a common coordinate system. Figure 2.7 shows a shaded image of a double-pyramid superquadric ($\epsilon_1 = \epsilon_2 = 2$) which has been twisted about one axis. To our knowledge there is no commercial CAD system employing superquadric primitives

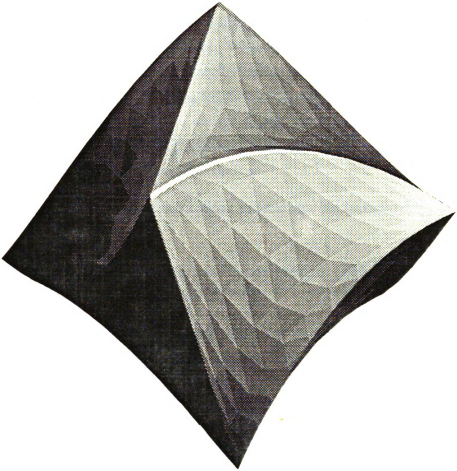

Figure 2.7: A twisted superquadric.

internally. Since most interest in superquadrics has come from vision researchers, this is not surprising [103]. Most existing work with superquadric primitives has dealt with fitting problems rather than the development of recognition systems employing superquadrics as an object representation scheme.

Feature-based Models

The solid modeling approaches discussed above describe objects in terms of primitives (primitive surfaces, curves, or volumes). While such descriptions capture the shape of an object, they do not explicitly describe ***design features*** such as holes, grooves, or notches. Recent research in CAD has led to the development of ***feature-based modelers*** [19, 5], which allow engineers to design objects at the feature level, avoiding the use of geometric primitives where possible. In addition, these modelers can maintain the semantics associated with features, such as

- a typical set of machining instructions to fabricate the feature (drilling, reaming, *etc.*),
- the local relationships among feature components (which part of the hole is hollow?), and
- constraints on manufacturability in terms of feature parameters (for example, we may only wish to allow 'hole' features with certain radius values, because our selection of drill sizes is limited).

Feature-based models will make downstream processing of mechanical designs (such as milling and assemblability analyses) less time-consuming; the inferences performed for those tasks are less complicated, as the reasoning starts at the feature level, rather than the local geometry level.

Feature-based models often employ B-reps as an internal representation. The explicit organization of local geometry into design features may allow vision systems to match on the features themselves, rather than their component entities. Park and Mitchell [89]) developed a visual inspection system employing feature-based CAD models.

GEOMOD Solid Modeler

In this section we describe GEOMOD [110], the solid modeler used in this dissertation to create 3-D models for our vision system. We also discuss the Initial Graphics Exchange Specification (IGES), the industry-standard format for interchange of geometric data. Comprehensive surveys of the state of the art in mechanical CAD are given in [93, 94, 95].

GEOMOD is a versatile solid modeling system using boundary representations.

1. Geometric primitives can be combined in a CSG-like fashion,
2. A 2-D profile can be extruded along a space curve or revolved around an axis,
3. A series of cross-sections can be assembled and a surface grown ('skinned') between them,
4. 3D points may be specified, linked into facets, and curved surfaces fit to the facets.

GEOMOD also offers deformation capabilities, including operations such as *bend*, *blend*, *stretch*, and *tweak*, to modify existing object geometry and produce different objects.

If the CSG-like assembly route is taken, the following primitives are available to the user: blocks, cones, cylinders, spheres, tubes, and quadrilaterals. The normal Boolean operations are available for aggregation of these primitives. The interface between CSG operations and sweeping, skinning, and faceting operations is seamless, because all surfaces are represented internally as nonuniform rational B-spline surfaces (NURBS) [114]. The NURBS surface form was given in Equation (2.1).

NURBS are quite powerful in terms of their representational capability. Unlike nonrational B-splines and most other parametric surfaces, they can represent quadric surfaces *exactly*. The complexity of obtaining intersection curves between these surfaces is simplified because of the uniform representation. These intersection curves are not necessarily exact, however; for example, the intersection between two quadrics is a fourth-degree algebraic curve [76], and may not admit an exact parametric representation. However, an approximating parametric spline curve of any desired accuracy can be easily obtained.

CAD Output: The IGES standard The CAD user community has largely agreed on a data interchange format for CAD systems, allowing designs created on one CAD system to be used by others. This format, known as the IGES standard [85], defines standard design entities for 2D drawings and 3D objects, as well as electrical diagrams, finite-element definitions, and other manufacturing quantities. We concentrate our attention on that portion of the IGES standard devoted to the description of 3D geometry.

Within IGES, 3D solids are specified as lists of the following geometric primitives:

- Lines, circular arcs, conic sections, parametric cubic spline curves, and NURBS curves.

- Planes, ruled surfaces, tabulated cylinders, surfaces of revolution, bicubic parametric spline surfaces, and NURBS surfaces.

Several of these primitives will be described in detail in Section 2.4.1.

Many CAD software products advertising 'IGES compatibility' actually implement only a subset of the IGES specification. For example, GEOMOD will write IGES files, but it only describes objects in terms of a subset of the available primitives. An IGES file can also be read by GEOMOD, but some primitive types are not accepted by the system.

Alternative Representations in IGES; Invisible Primitives The IGES format can often specify objects in two different ways. Lines, circular arcs, parametric cubic curves, planes, natural quadrics, and surfaces of revolution can be expressed either as special cases (called the *analytic* form by GEOMOD), or as NURBS (the GEOMOD *rational* form). A NURBS description is more powerful (*i.e.* not all objects composed of piecewise-NURBS surfaces bounded by NURBS curves can be represented exactly using only planes, natural quadrics, and surfaces of revolution), but intuitive geometric information about the object (*e.g.* symmetry axes, radii, and surface type) is not explicit in the NURBS forms.

For example, consider a cone C, of height 1, with its axis coincident with the y-axis, its vertex located at $(0,1,0)$, and a radius of 1.0 at the base (the xz-plane). How can this curved surface be represented within the IGES specification?

One representation for this surface is the NURBS form, which, in the context of Equation (2.1), has

- $n = 8, m = 2$,
- eighteen control points $\mathbf{P}_{ij}$,
- eighteen control point weights h_{ij},
- twelve knot values in one parameter, and
- four knot values in the other parameter.

Given this information, how do we decide that the parametric surface is a right circular cone? Once we have decided that it is conical, how do we obtain the geometric parameters (namely, the coordinates of the vertex, the vertex angle, and the orientation of the axis)?

In addition to the complicated NURBS description, we can also describe the cone as a surface of revolution: a linear *generatrix*

$$\mathcal{G} = \overline{\mathrm{p}_1\mathrm{p}_2}, \quad \mathrm{p}_1 = (1,0,0), \quad \mathrm{p}_2 = (0,1,0),$$

revolved about a linear axis

$$\mathcal{A} = \overline{\mathrm{p}_3\mathrm{p}_4}, \quad \mathrm{p}_3 = (0,0,0), \quad \mathrm{p}_4 = (0,1,0).$$

Note that the axis of the cone is not uniquely determined; either endpoint (or both) could be 'extended' to lengthen the axis without changing the revolved surface. This representation of the cone surface is called the *analytic* representation by GEOMOD.

In the analytic form of this cone, $\mathcal{G}$ and $\mathcal{A}$ are geometric primitives but are not visible edges on the object; they are referred to as *invisible* entities in the IGES specification. Despite the fact that they cannot be sensed (and therefore should not be matched to scene entities during the object recognition process), these invisible entities convey valuable information about surface shape (*i.e.* the axis of symmetry, the vertex location and vertex angle, *etc.*).

Analytic IGES: Motivation In constructing our object representation, we start with the *analytic* IGES output from the CAD software. Lines, circular arcs, planes, and surfaces of revolution are described explicitly in terms of their geometric parameters (see Section 2.4.1). This representation allows us to easily extract important attributes and relations from the CAD model. More general surface types (*i.e.* NURBS surfaces) are not handled by our current modeling system.

Our use of the analytic IGES form rather than the rational form (in which *all* primitives are expressed as NURBS curves or surfaces) is motivated primarily by some computational considerations. It is possible (in principle, at least), to recover the surface type and geometric parameters from the NURBS description. Let us examine the recovery process associated with the case of a NURBS description of a quadric surface. The implicit equation of a quadric surface is

$$f_{quadric}(x,y,z) = a_1x^2 + a_2y^2 + a_3z^2 + a_4xy + a_5xz + a_6yz + a_7x + a_8y + a_9z + a_{10} = 0.$$

Given a set of points $\{(x_i, y_i, z_i), i = 1, ..., n\}$ (which can be obtained by sampling the parametric NURBS form), a quadric can be fit (if $n = 10$) by solving a system of ten equations in ten unknowns. However, the finite precision of the floating-point numbers can adversely affect the quality of a solution to the system. The traditional method for overcoming this difficulty is to perform regression: insist $n > 10$ and obtain the estimate $\hat{f}_{quadric}$ which minimizes

$$E^2 = \sum_{i=1}^{n} \hat{f}^2_{quadric}.$$

Invariant-based methods for classifying quadrics from the set of coefficients obtained via regression have been published [49], but planar, cylindrical, and conical surfaces require that an invariant of the parameter vector be tested against zero. When the sensor data are contaminated by noise, these tests will *always* fail unless some thresholds are established. The behavior of these invariants with respect to noise has not been studied, making the choice of such a threshold an open problem.

IGES as a representation We wish to stress the following point: IGES was intended to be used as a *data interchange standard*, not an object representation method. In our work, we are treating the IGES information as a description of the 3D object, and limit our attention to a subset of the IGES 3D curves and surfaces.

One of the shortcomings in the IGES standard deals with the completeness of representations for solid objects [120]. The mathematical notion of completeness is given in Section 2.2; briefly, a representation is complete if it corresponds to only one physical object. While the latest revision of the IGES standard (version 4.0) allows 3D objects to be specified as CSG trees (which are complete) [85], the descriptions employing curve and surface primitives alone are not complete. The primary shortcoming is the lack of explicit connectivity information. In most solid modelers, connectivity [25] is expressed in a graph on the set of surfaces, edges, and vertices. Since we are limited in our work to the 'partial' B-rep produced by IGES, the lack of connectivity is a concern. 3D object recognition strategies make extensive use of surface and edge connectivity during the matching process. Once a range or intensity image has been segmented into regions, we *know* that adjacent image regions sharing a crease edge are connected. Many object recognition systems in the literature perform matching by establishing a correspondence between a 'kernel' image feature (such as a long dihedral edge or a large region) and a model feature, and then 'grow' the match to adjacent model and image features to which the kernel is connected. If connectivity is not explicitly represented in the model, it must be *re-inferred* when the vision models are being constructed. While we have made some attempts in this direction [40], our need for viewpoint-dependent information provided an opportunity to extract topological information reliably using additional information provided by the solid modeler. This process is described later, in Section 2.4.5.

2.3.2 Representations Used in Vision Systems

The following representations have been used primarily by researchers in computer vision.

Sweep Representations

One popular method for shape representation which is suitable for objects whose volumes are *swept* by *generating curves* with respect to one or more axes. We will concentrate on objects which are representable as a single such swept volume, and stipulate that several such volumes are required to represent 'interesting' objects such as industrial parts. In the vision community, sweep representations have been well-known as *Generalized Cylinder* (GC) representations [1, 86, 22][2].

A single swept volume is characterized by

- an axis (which may be straight or curved); and
- a cross-sectional area which sweeps out the volume as it travels along the axis.

[2] Many CAD systems offer sweeping as a mechanical design option, but few systems employ it as an internal representation.

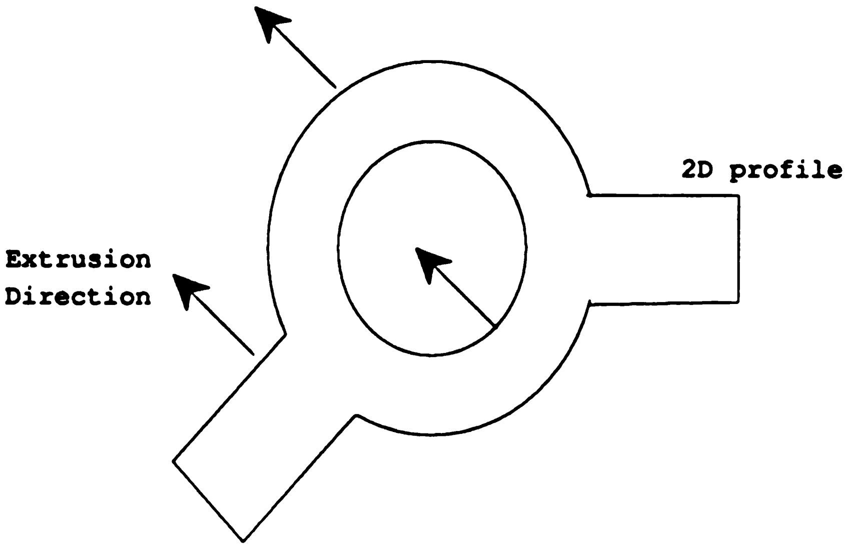

Figure 2.8: A sweep representation of the example object.

The cross-section need not contain the axis, and it may undergo translation, rotation, scaling, or all three as it is being swept. Most researchers, however, assume that the cross-section is planar and that its plane is perpendicular to the axis curve at all times. Figure 2.8 shows that the example object can be represented as a linear extrusion of a 2D profile. Here, the cross-section is indeed planar, and the axis is straight. It is interesting to note that some sweep operations correspond closely to industrial fabrication operations. The example above showed linear extrusion. If the cross-section was also rotating, we would obtain a twisted object (*e.g.* a gear). If the shape is constant but the scale changes monotonically, the result is a constant-taper object.

An object with n swept subparts might be represented as follows:

$$\mathcal{R}_{sweep} = \{S_1, S_2, \ldots, S_n\},$$

with each of the S_i being a pair

$$S_i = \{\alpha_i, A_i\}, \text{where}$$

α_i is an expression (parametric or otherwise) for the axis, and A_i is the area as a function of position on the axis.

Geons: Qualitative Representations

Most solid representations used in CAD or vision systems are *quantitative* in nature; primitive surfaces (or volumes, *etc.*) are specified in terms of numerical parameters. Such detailed descriptions are necessary in fabrication, simulation, and some vision tasks (such as metrology). However, if we are performing a visual recognition task and the objects to be recognized can be distinguished by examining some qualitative features of the segmented primitives, representations which capture *only* those variations might offer advantages in processing [3]. The idea of qualitative representation was proposed by Biederman [12], who developed a catalog of 36 *geons* (geometric ions) with each member of the catalog having a unique set of four qualitative features:

- *Edge*: straight or curved;
- *Symmetry*: rotational/reflective, reflective, asymmetric;
- *Size Variation*: constant, expanding, expanding/contracting;
- *Axis*: straight or curved.

Figure 2.9 shows three geons with their qualitative features. Most interesting 3D objects would be composed of a number of connected geons. Biederman proposes an edge-based procedure for segmenting images into their geon components, using nonaccidental alignments and concavities to locate the places where geons join. Over 154 million qualitatively different objects can be constructed from three geons and inter-geon relationships, considering all possible relative sizes, arrangements, and affixments of the three components [12]. Therefore, the geon models are a very versatile coarse description of 3D objects. Bergevin and Levine [6] have developed a model-building system which attempts to form geon models from 2D line drawings.

While the geon representation has intuitive appeal, the lack of quantitative information limits its usefulness in environments where discrimination between qualitatively similar, but quantitatively different objects is performed. If an assembly line is making a 'family' of parts which differ only in scale, they would have identical geon representations, making discrimination between the different sized items impossible without additional (quantitative) information.

[3] However, a qualitative representation cannot (in general) be used to synthesize an image of the object.

:

I

se

graph

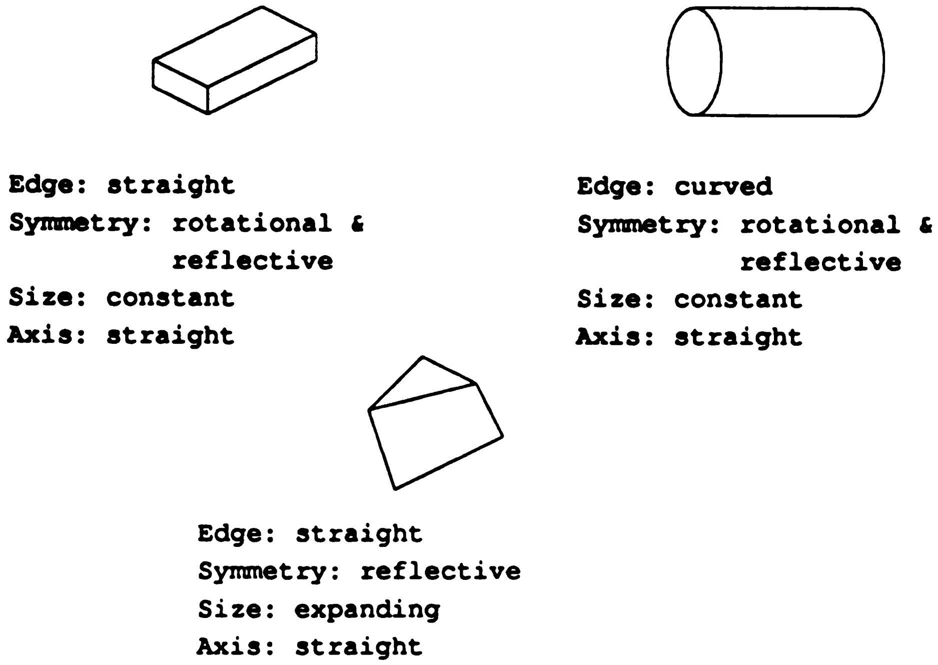

Figure 2.9: Three geons and their qualitative features

Multi-View Representations and Aspect Graphs

One method for describing a 3D object is to describe some or all of its possible 2D projections. These projections are often known as *2D* or *view-centered* models. In some applications, the number of stable positions the object can assume on a supporting surface is small. Suitable features calculated from those views can be used in recognition via statistical, syntactic, or graph-theoretic approaches [31]. In most of these systems, separate views of the same object are stored separately, as if they were actually different objects. Dudani *et al.* [34] developed one of the earliest recognition systems employing multiple views. The model set consisted of 132 images of six different aircraft. Each view was represented by fourteen numerical features, which were invariant moments calculated from the boundary and silhouette of the object in the given view. The matching module considered each database view independently of the others.

Other representation methods attempt to combine all these viewpoint-specific models into a single data structure. There has been a recent burst of activity among several computer vision researchers seeking algorithms which compute the *aspect graph* of both polyhedral and nonpolyhedral objects [35, 104]. The aspect graph

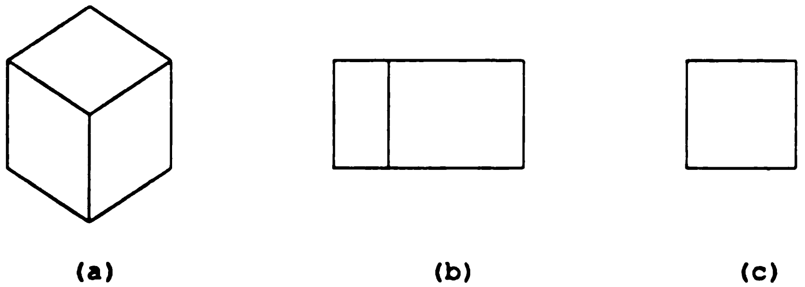

Figure 2.10: Qualitative changes in the line drawing of a cube.

summarizes the qualitative appearance of an object from every possible viewpoint. The meaning of 'qualitative' differs from researcher to researcher; often, the qualitative appearance involves the relative positions of junctions between creases on the object surface and its silhouette (i.e. the 'line-drawing' topology). Figure 2.10 shows three qualitatively different line drawings of a cube. Figure 2.10(a) shows a 'general' view, in which three surfaces are visible, along with seven vertices. When the cube is rotated so as to obscure one of the faces (Figure 2.10(b)), one vertex disappears along with the edges incident on it. A further rotation eliminates an additional face and two vertices (Figure 2.10(c)). In this discussion, we will define a 'qualitative' change as a topological change in the line-drawing of the object [4].

Discussion of a representation based on views of an object requires a definition of 'view' along with the number and type of parameters which define it. In the context of intensity images, the type of projection assumed in the image formation process must be specified. Under the perspective imaging model, the absolute distance (the *standoff*) between the object and the sensor distorts the position and orientation of objects in the image. Many authors avoid the problems of perspective distortion by assuming an orthographic projection model. Perspective distortion is most pronounced when the sensor is very close to the object being sensed; if we can guarantee that the standoff is fairly large in relation to the size of the object being viewed, the orthographic assumption is not unreasonable to make. Figure 2.11 illustrates the difference between orthographic and perspective projections in terms of visible sides of a 2D object. The sides of the object are labeled 1 through 6 and are drawn with wide dark lines. The sensor can be positioned anywhere outside the object. The gray

[4] An alternative notion of qualitative appearance was proposed by Ikeuchi and Kanade [62]; they define two views as qualitatively identical if the same faces are visible in both.

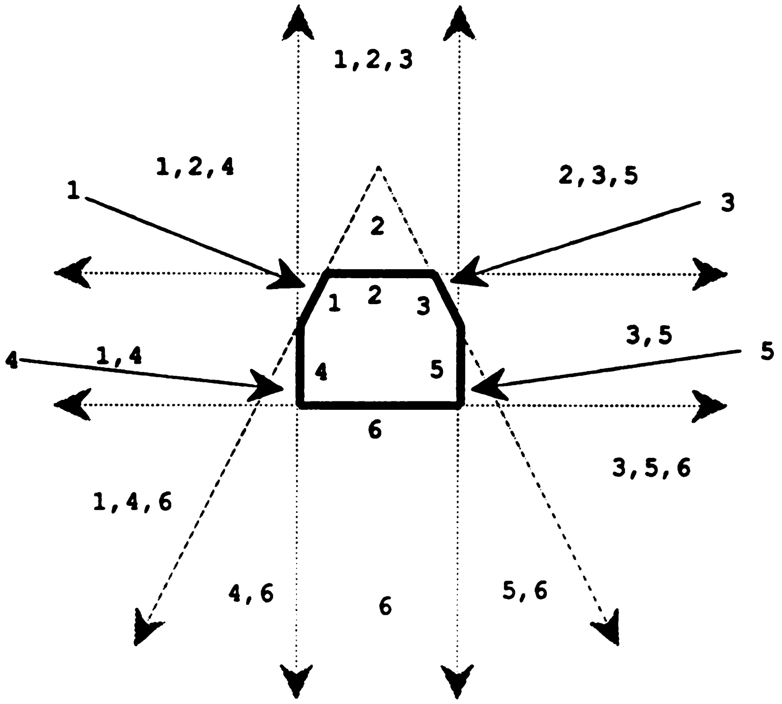

Figure 2.11: Effects of standoff on visibility of a 2D object

lines divide the plane into regions; each region is labeled with the visible faces of the object in that region. Notice five small triangular regions associated with faces 1, 2, 3, 4, and 5. If the (1D) sensor is placed within those regions, only the associated face will be visible. At larger standoffs, these effects diminish and the orthographic projection is a reasonable assumption.

Unlike intensity images, range images are not projections; therefore, the image-plane positions of image artifacts will not undergo perspective distortion. In a practical sense, this means that we need not worry about a change in standoff producing a change in the image. The set of parameters defining a view has three members, namely the three components of a rigid rotation that we impose on the object to orient it in the field of view of the sensor. Most researchers ignore rotations about the viewing direction, since such rotations will not affect its qualitative appearance. Therefore, a view is specified by two angles, or (alternatively) a 'viewing direction' expressed as a unit vector. The space of view directions lies on a sphere of unit radius called the *viewsphere*.

With the constraints specified above, we are now in a position to define the aspect graph structure. The *aspect graph* of an object has the form

$$\mathcal{R}_{aspect} = \{\mathcal{V}_{aspect}, \mathcal{E}_{aspect}\}.$$

Vertices $v \in \mathcal{V}_{aspect}$ represent open connected subsets of the viewsphere from which the qualitative appearance of the object is the same. Edges $e \in \mathcal{E}_{aspect}$ between vertices represent so-called *visual events*, changes of viewpoint in which the qualitative appearance changes. Visual events occur at viewpoints on the boundary of two or more regions of the viewsphere associated with vertices. When the object is viewed from viewpoints lying on these boundaries, the object's qualitative appearance is undergoing a transition; these viewpoints are called *accidental.* Figure 2.12 shows an aspect graph for a cube. The labeling of the cube's faces is also shown in the figure. At each node in the graph, between one and three faces are visible. Traversing an arc between nodes corresponds to a change in viewpoint that either occludes a face or makes one visible.

Formal approaches to aspect graph construction seek an exact partitioning of the view sphere. Such techniques analyze some form of geometric model for the object and obtain the transition boundaries mentioned above using a 'catalog' of allowable visual events for the particular object. A thorough exploration of the competing techniques is beyond the scope of the present discussion. However, the two major drawbacks of the formal techniques for aspect graph construction are

- Large time and space complexity (for a non-convex polyhedron with n sides and assuming orthographic projection, one existing algorithm [43] requires O(n^8) time and O(n^8) storage to compute the aspect graph)[5]; and

- A lack of implemented algorithms for computing the aspect graphs of non-polyhedral objects. Recent work has highlighted some formidable numerical problems that must be solved before reliable implementations will become available [104].

Additionally, the current work on 'formal' aspect graphs has emphasized their construction from geometric models, rather than their use in recognition. Once the

[5] The construction of aspect graphs would typically be performed off-line (not during recognition). Searching (at recognition time) the data structure produced by these algorithms might be time-consuming, however.

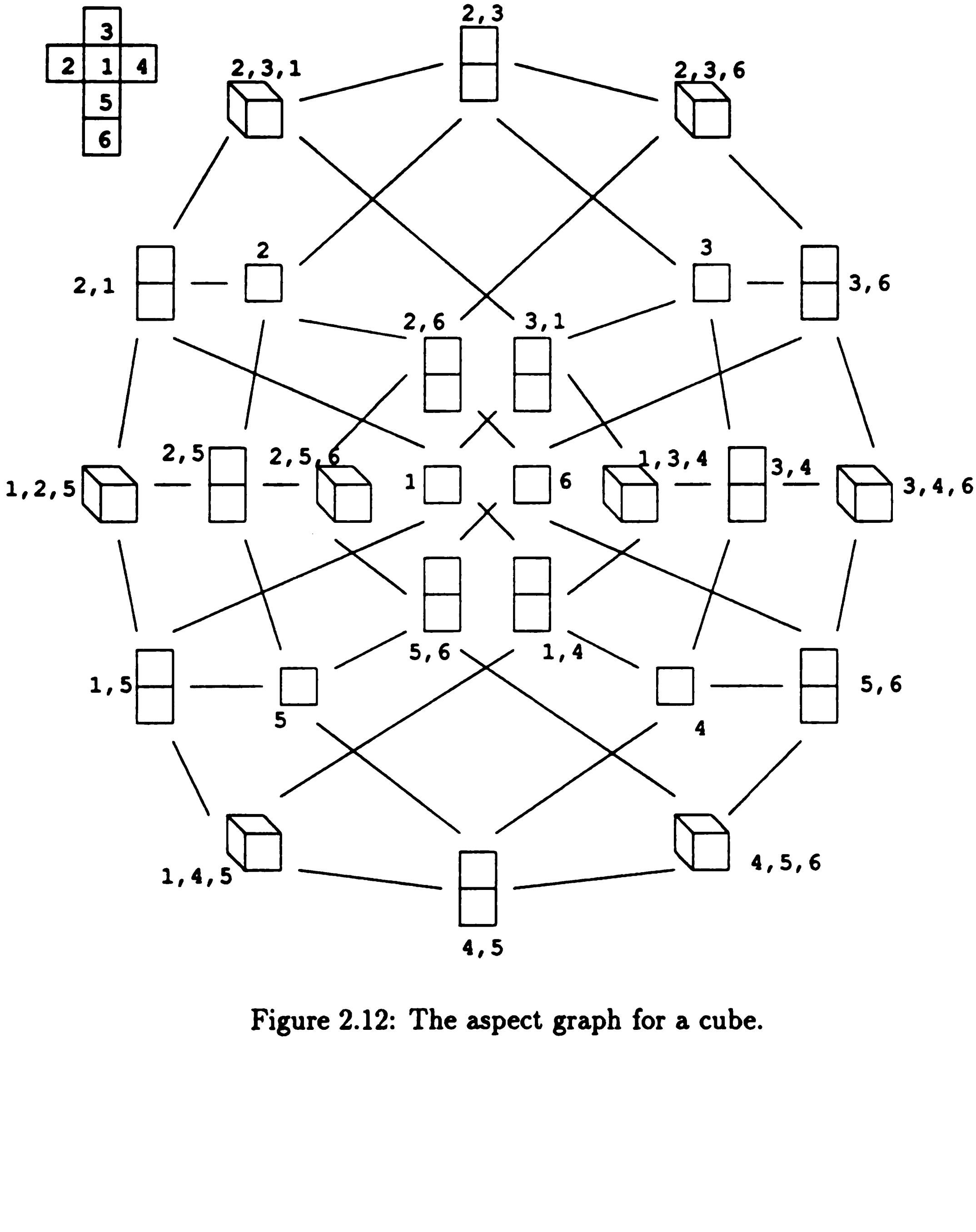

Figure 2.12: The aspect graph for a cube.

difficult problems of actually building the aspect graph are solved, the important work on recognition must take place.

Some researchers have taken a pragmatic approach to view-centered representation [62, 72, 26, 67]. Rather than obtaining an exact partitioning of the viewing sphere for each object, a *discrete sampling* of the viewsphere is taken, images of the object are synthesized for each sample point, and some appropriate features are computed from each image (see Section 2.4.5). Optionally, sets of viewpoints producing similar feature lists are merged into a view class and a representative image is stored with that class. The pragmatic approach to viewpoint-dependent representation was adopted in this dissertation; our modeling system incorporates such view-dependent features as sensed area and simultaneous visibility into a viewpoint-independent model, providing additional information that has proven useful to the recognition procedures.

Chen and Stockman [29] have introduced a new object representation scheme which associates surface type with contour information. This model is built from several range images of the object. Each edge pixel in the image is analyzed to extract its type (silhouette edge, interior jump edge, or crease edge), and the edge points are grouped into contours of the same type. The label for each contour segment is then augmented with information about the surfaces on either side of the edge (unknown, planar, convex, concave, or saddle). Hence, each contour is identified by a triple of attributes, and is called an *object wing.* Figure 2.13 shows the contour and surface types for three objects which have identical line drawings: a bowl, a half grapefruit, and a clam. The combination of line and surface labelings produces a different wing description of each object. Chen and Stockman show that with five surface types (null, planar, concave, convex, saddle) and four contour types (silhouette, jump, convex crease, concave crease), only 34 of the 100 triples containing two surfaces and one contour correspond to realizable wings. One important feature of the wing representation is that it is designed to accommodate errors in segmentation. This is a very desirable property, because the extraction of edges (especially crease edges) from range imagery is quite difficult.

In our object modeling system, relational graphs constructed from B-reps are augmented with some surface-type-specific information from a uniform viewsphere sampling. No grouping of views into view classes is performed.

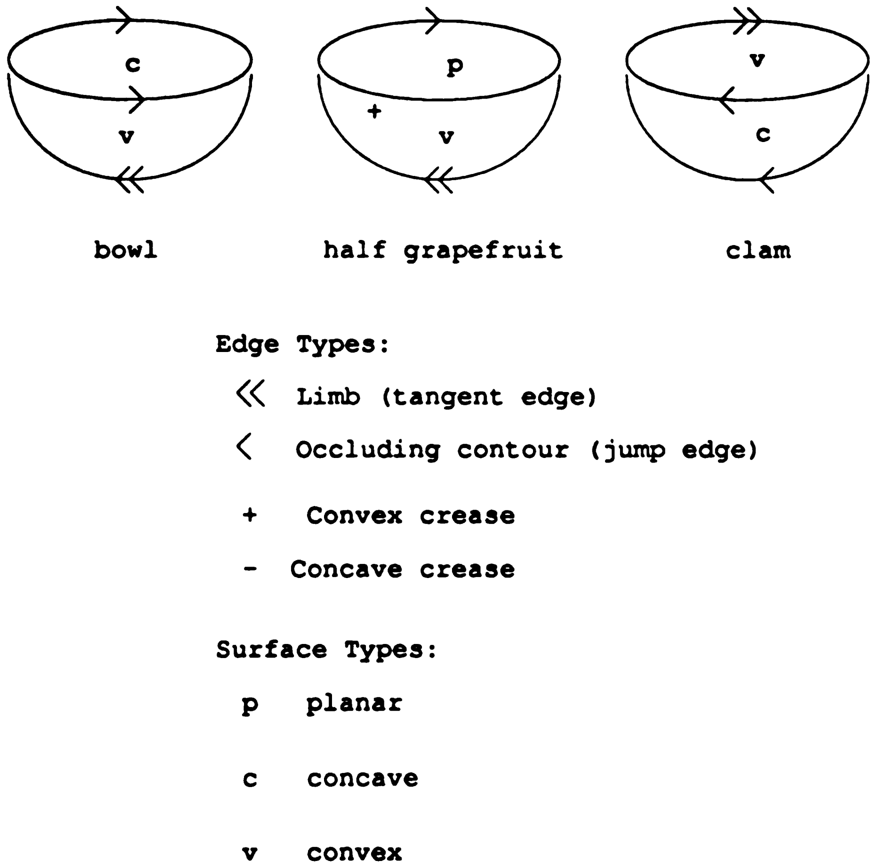

Figure 2.13: Contour and surface types for three objects with identical line drawings.

2.4 CAD Models to Relational Graphs

What information is *explicitly* stored in the IGES object description, and what additional information about the object should be stored explicitly to improve and speed up the object recognition process? How is this information arranged? In this section, we describe the geometric primitives produced by our CAD system and the attributes of binary relations between primitives that we derive. The transformation applied to object descriptions expressed in the IGES form produces our vision model, stored as an *attributed relational graph*

$$G = < \mathcal{V}, \mathcal{E} >,$$

with vertex set $\mathcal{V}$, containing geometric primitives and attributes:

$$\mathcal{V} = \{(v_i, a_i) | i = 1, ..., n\},$$

$$v_i \in \{\text{circular-arc, line, parametric-spline-curve, plane, surface-of-revolution}\},$$

where a_i is an attribute set associated with v_i; here, attributes are the geometric parameters of the particular curve or surface.

The edge set $\mathcal{E}$ contains attributed binary relations between nodes. Binary relations can be both unidirectional (*e.g.* the inheritance relation between a surface of revolution and its linear axis) and bidirectional (*e.g.* the angle between two lines).

Our relational graph contains redundant information. The computational burden of relational graph construction is *not* incurred at object recognition time, however; the transformation of IGES models to relational graphs need only be applied when a new object definition is created and the corresponding vision model is needed. Each object is handled separately. So, addition of an object to the database does *not* change the representations of existing models.

Our choice of geometric features (attributes and relations) is motivated by sensory considerations. We attempt to preserve explicitly the geometric information which most concisely describes the objects in the database: orientation vectors, radii, and surface type labels. Additionally, these quantities can be computed directly from range data: radii and orientation vectors can be found from the principal curvatures and their directions (see Chapter 4). The vision models we construct contain both viewpoint-independent features such as surface intrinsics (radii) and relative quantities (inter-patch orientations), and viewpoint-dependent features (*e.g.* patch areas).

Th

cudin

Then,

Final

depen

depen

tion.

2.4.

In thi

graph

tion o

Lines

A. L

a sta

The

is a s

segm

B. C

ordi

and

into

mat

-

-

-

-

The geometric inferencing performed here first examines every IGES entity (including visible entities) in the CAD model individually to calculate node attributes. Then, all pairs of visible entities are processed, and binary relations are produced. Finally, the set of synthetic views is generated and analyzed to produce the view-dependent features for each model surface. In terms of processing time, the view-dependent feature calculations dominate the time needed for vision model construction.

2.4.1 Node Attributes

In this section we define attributes for geometric primitives used in the relational graph. These attributes are either stored explicitly in the *analytic* IGES representation or calculated by our inference system.

Lines and Curves

A. Line segments. In the IGES specification, a line segment $\mathcal{L}$ is characterized by a starting point and an ending point.

$$\mathcal{L} \;=\; \overline{\mathbf{p}_1\mathbf{p}_2}.$$

The segment's length

$$\|\mathcal{L}\| = \|\mathbf{p}_2 - \mathbf{p}_1\|$$

is also computed and stored as a node attribute for $\mathcal{L}$. Henceforth, we will refer to line segments as lines (with the implicit understanding that we actually mean segments).

B. Circular Arcs. In IGES, a circular arc $\mathcal{A}$ is specified in its own (arc-centered) coordinate system (x_a, y_a, z_a), in which the plane of the arc is parallel to the $x_a y_a$-plane, and displaced from it along the z_a axis. This 'canonical' curve is then transformed into model-centered coordinates by a 3×4 transformation (rotation and translation) matrix. The five quantities in the IGES description of the arc are:

- the parallel distance z_t along the z_a axis between $\mathcal{A}$ and the $x_a y_a$-plane;
- the location (x_{ac}, y_{ac}) of $\mathcal{A}$'s center in arc-centered coordinates;
- the location (x_{as}, y_{as}) of $\mathcal{A}$'s starting point in arc-centered coordinates;
- the location (x_{ae}, y_{ae}) of $\mathcal{A}$'s ending point in arc-centered coordinates; and

-

Our

tive:

-

-

-

-

-

C.

any

spli

divi

in e

low

Th

pre

pol

int

fac

GE

D.

tat

- a transformation matrix $\mathbf{R}_{\mathcal{A}}$ mapping arc-centered coordinates to model-centered coordinates.

Our system computes the following additional attributes for each circular arc primitive:

- the starting and ending angles of rotation in the $(x_a y_a)$-plane;
- the fraction of a circle produced by the rotation;
- the coefficients of the plane (in model-centered coordinates) containing the arc;
- the radius of the arc; and
- the location of the arc's center in model coordinates.

C. Parametric Spline Curves. In the analytic IGES files produced by GEOMOD, any model curve which is not linear or circular is expressed as a parametric cubic spline curve $\mathcal{C}$, possibly with multiple segments. The range of parameter values is divided into subsets by *break points*, and a different parametric cubic curve applies in each segment (curves whose parameter ranges are adjacent are required to join, however). These curves have many parameters. The ones of interest to us are:

- a 'planarity' flag (curves with this flag set lie in a plane);
- the number n of curve segments;
- the $n+1$ breakpoints $\{t_1, ..., t_{n+1}\}$ in parameter space; and
- coefficients of $n+1$ vector-valued univariate polynomials, describing the (x, y, z) coordinates of the n segments defined by the breakpoints.

The curve parameter t lies in the interval $[t_1, t_{n+1}]$. One additional parameter is present when $\mathcal{C}$ is planar. In that case, the IGES specification requires that the z polynomial must be constant-valued. The curve thus defined needs to be transformed into the model-centered coordinate system using a 3×4 transformation matrix. In fact, such a transformation can be supplied even for nonplanar curves; however, the GEOMOD software does not produce transformations for other than planar curves.

D. Composite Curves. The IGES specification allows for a hierarchical representation of object curves. A composite curve entity is simply a list of *subcurves*, each of

which
are re
second
a plan
the cu
in the

Surfa

A. P

-

-

It is
the o
consi
boun
allow
para
obje
plan

I
when
boun
2D
attri
matr
with
coeff
cente

which can be any IGES curve type, such as the three mentioned above. The curves are required to 'join', *i.e.* the endpoint of the first subcurve is the beginning of the second, *etc.* In our models, composite curves are used only as the 'bounding' curve of a planar surface (discussed below). We compute no additional attributes for composite curves at present. Much of the geometric information about the curve is embodied in the subcurves, and can easily be retrieved from them if required.

Surfaces

A. Planes. An IGES planar entity $\mathcal{P}$ contains the following information:

- Coefficients: four scalars a, b, c, and d, such that points in $\mathcal{P}$ satisfy

  $$ax + by + cz = d.$$

  We choose the signs of the coefficients so that (a, b, c) is normal to the plane and points out of the solid object bring described.

- Bounding curve: a pointer to another IGES entity which describes a closed curve that encloses $\mathcal{P}$.

It is important to note that the bounding curve does *not* (in general) coincide with the object edges which bound the planar surface under consideration. For example, consider one of the two planar end patches on a right circular cylinder. The 'physical' bounding curve for this planar patch is a circular arc, but the IGES specification allows the bounding curve to be piecewise-linear (*e.g.* a rectangle), or even a closed parametric curve. In other words, the bounding curve need not correspond to any object edges; it suffices that the bounding curve enclose the 'valid' portion of the plane.

It is often useful to reduce the dimensionality of geometric computations to 2D when planes are involved (for example, it is easier to decide that a point lies in the bounded portion of the plane if the point and bounding curve are transformed into 2D than to do the computations in 3D directly). For this reason, we calculate two attributes for each model plane. First, we find a 4 × 4 homogeneous transformation matrix $\mathbf{R}_{\mathcal{P}}$ which maps 3D points into a 'plane-centered' coordinate system (x_p, y_p, z_p), with $z_p = 0$ on the model plane. This transformation is obtained from the four coefficients a, b, c, and d. A bounding box for the valid region of the plane in the plane-centered coordinates is also calculated. The IGES bounding curve is transformed into

(x_p, y_p, z_p) coordinates and sampled. The bounding box of this sampled curve is stored in the vision model.

Finally, each IGES plane has a list of areas corresponding to the viewpoints on the uniformly sampled viewsphere. The methods for synthesizing images and estimating the visible area appear in Section 2.4.5.

B. Surfaces of Revolution. For a surface of revolution $\mathcal{S}$, the following properties are given explicitly in the IGES description:

- a pointer to the axis of revolution $\mathcal{L}_\mathcal{R}$ (a line entity),
- a pointer to the *generatrix* $\mathcal{G}_\mathcal{R}$ (the line or curve which is revolved about the axis to 'sweep' out the surface), and
- the starting and ending angles of rotation for the sweeping operation.

Additional attributes calculated for the vision model include

- a more precise surface classification:
  1. *cylindrical* if $\mathcal{G}_\mathcal{R}$ is linear and parallel to $\mathcal{L}_\mathcal{R}$;
  2. *conical* if $\mathcal{G}_\mathcal{R}$ is linear and not parallel to $\mathcal{L}_\mathcal{R}$;
  3. *spherical* if $\mathcal{G}_\mathcal{R}$ is a circular arc and $\mathcal{L}_\mathcal{R}$ passes through the center of the arc;
  4. *toroidal* if $\mathcal{G}_\mathcal{R}$ is a circular arc and $\mathcal{L}_\mathcal{R}$ does not pass through the center;
  5. *generic* if $\mathcal{G}_\mathcal{R}$ is nonlinear and noncircular.
- if the surface is cylindrical, the radius;
- if the surface is conical, the vertex location and vertex angle;
- if the surface is spherical, the radius and center location;
- the orientation of the axis.

The radius of a cylinder is given by the perpendicular distance between $\mathcal{G}_\mathcal{R}$ and $\mathcal{L}_\mathcal{R}$. The vertex angle of a conical surface of revolution is obtained from the scalar product of the orientation vectors of $\mathcal{G}_\mathcal{R}$ and $\mathcal{L}_\mathcal{R}$. The cone's vertex is located at the intersection of $\mathcal{L}_\mathcal{R}$ and $\mathcal{G}_\mathcal{R}$. If $\mathcal{S}$ is spherical, the center and the radius are inherited

Table 2.1: IGES representation and inferred quantities for vision models

Entity	Analytic IGES representation	Additional Features in Vision Model
Line	starting and ending points	length
Circular Arc	canonical representation transformation	angles fraction of circle plane coefficients radius, location
Parametric Spline Curve	planarity flag number of segments breakpoints polynomials	-
Composite Curve	pointers to subcurves	-
Plane	coefficients bounding curve	canonical transformation canonical bounding box visible area list
Surface of Revolution	axis generatrix angles of rotation	type orientation of axis type-specific attributes visible area list

from $\mathcal{G}_{\mathcal{R}}$. In addition, the visible area of $\mathcal{S}$, indexed by viewpoint number, is calculated for each point on the uniformly sampled viewsphere.

The radii of spherical and cylindrical surfaces of revolution are often highly informative features for object recognition. In addition to storing these radii along with the corresponding nodes in the relational graph, we also compute a global list of radii for each model. Examination of this when processing an input image allows models with dissimilar curved surfaces to be rejected by the object recognition module.

Table 2.1 summarizes the geometric primitives provided in the IGES description and inferred by our system. In many manufacturing environments, additional attributes not dealing with the geometry of the object surfaces (such as surface finish, paint color, a fabrication script, or material type) accompany the geometric attributes. If these features were considered useful for object recognition, they could also be expressed as node attributes.

2.4.2 Binary Features

In this section we describe the binary features derived from examinations of *pairs* of IGES geometric primitives. These features are stored as attributed arcs between

the
fal
whic

Ori

One
with
of t
cer
an
en

pa
en
ori

wh
rel
int
the

fro
rela

Table 2.2: Geometric Primitives and Orientation Attributes

Primitive	Orientation Parameter
Line	Direction vector
Circular Arc	Normal to plane of arc
Parametric Spline Curve	none
Plane	Normal Vector
Surface of Revolution	Axis Direction

the attributed nodes in the relational graph representation. These binary relations fall into three categories: *orientation relations*, *proximity relations*, and *miscellaneous relations.*

Orientation Relations

One local, intrinsic object property that has been used extensively in 3D object recognition systems is *orientation*; specifically, the relative orientation of object faces. One of the reasons we have focused on a subset of the IGES representation containing only certain geometric primitives is that most of those primitives contain an orientation attribute of some sort. Table 2.2 lists the orientation attributes for each of the IGES entities in our system.

An *orientation relation* is produced by our geometric inference engine for each pair of geometric primitives for which an orientation attribute exists. In other words, entity pairs containing one or two parametric spline curves produce no relation. The orientation relation for two distinct primitives prim_i and prim_j has the form

$$\theta_{ij} = \texttt{angle}(\text{prim}_i, \text{prim}_j),$$

where θ_{ij} is the angle between their orientation attributes. Thus, the orientation relation between two lines $\mathcal{L}_1$ and $\mathcal{L}_2$ is simply the angle between them. A less intuitive example is the angle relation between a plane and a surface of revolution. If the normal to the plane is parallel to the axis of the surface, then $\theta = 0$.

In the section on node attributes, we discussed a global feature list constructed from the radii of curved patches. A similar list is constructed from the orientation relations. As above, the purpose of this list is to provide evidence for a certain model's plausibility at recognition time. For example, an angle between two planar surfaces observed in the scene can be used to select models for matching in the recognition module. An observed angle of 45° might cause us to include in the model subset being searched all models with such an angle between two patches. Unfortunately, not all orientation relations are equally salient for recognition. Some of our models contain a large number of 90° and 0° angles between faces. Observation of such an angle in the scene is not especially informative, as it implies that we should search *all* models with such an angle. The extraction of salient features (unary and binary) from models is an area of future research.

Proximity Relations

Proximity relations summarize the distance between entities. For pairs of curves (lines, circular arcs, and parametric spline curves), we instantiate a `distance-interval` relation and fill its two slots with the maximum and minimum distances between the curves. The only proximity relation between a curve and a surface primitive is `coincident`: this relation is added if the curve lies entirely within the *finite* surface primitive. If two model surfaces are adjacent (*i.e.* they share a visible curve), an attributed `adjacent` relation is added between the two surfaces, with the shared edge appearing as an attribute in the relation.

It is possible, in principle, to calculate `distance-interval` relations for edge-surface and surface-surface pairs as well as edge-edge pairs. Sampling of the primitives is not an appropriate strategy because of the large amount of computation time that would be consumed in just evaluating distances. Analytic techniques can be applied, at least in theory; the minimum and maximum distance between two points can be calculated by finding the minimum distance using a direct solution to a least-squares distance criterion [82]. Implementation of this solution presents practical difficulties, as mentioned below.

Consider the process of finding the minimum and maximum distance between two parametric cubic spline curves

$$\alpha_1(u) = (x_1(u), y_1(u), z_1(u))$$

and

$$\alpha_2(v) = (x_2(v), y_2(v), z_2(v)).$$

The

by

whi

val

min

clos

tech

the

por

met

surf

four

reas

curv

Mis

The

as e

rela

circ

the

rela

ran

Bes

in-

geo

enti

enti

cally

in ty

The squared Euclidean distance between the curves (as a function of u and v) is given by

$$D(u,v) = (x_1(u) - x_2(v))^2 + (y_1(u) - y_2(v))^2 + (z_1(u) - z_2(v))^2,$$

which is a sixth-order function of two variables. Finding the minimum and maximum values of D involves locating values (u_i, v_i) where $\frac{\partial D}{\partial u}$ and $\frac{\partial D}{\partial v}$ are zero, *and* the determinant of the Jacobian is positive. There will typically be several such values. Since closed-form solutions for roots of fifth-order polynomials do not exist, some numerical techniques must be employed. Each value must be checked against the domains for the parameters u and v to discard distance measurements made outside the 'valid' portion of the curves. The analogous situation for a parametric curve and a parametric surface adds an additional parameter. If the distance between two parametric surfaces is to be minimized, the minimum and maximum of a *sixth-order function of four parameters* has to be found. In our present work, we felt that sampling provided reasonable results for curve-curve pairs, and coincidence and adjacency relations for curve-surface and surface-surface pairs were sufficient.

Miscellaneous Relations

There are a few binary relations between geometric primitives which we categorize as either *identity*, *containment*, or *correspondence*. One example of a containment relation is coincidence between a pair of planes. Suppose our model contains a visible circular arc and a visible plane which is coincident with the arc (we do not require the bounded plane to contain the plane of the arc). The

`coincident-with` (plane$_i$, plane$_j$)

relation between these two primitives means that they have the same geometric parameters, but the physical surfaces on the model are disjoint. If a model curve lies within the two-dimensional bounding box calculated for a planar primitive, a `in-plane-bbox` relation is present. Identity relations express either inheritance of geometric information (*e.g.* a surface of revolution inherits the orientation of the line entity used as its axis), or a complete correspondence between curves (*e.g.* two line entities with coincident endpoints have an `identical` relation between them). Finally, correspondence relations are used to flag similar surfaces of revolution (similar in type, or in type and radius).

Relatic
angle
coinc
copla
dista
ident
in-pl
in-ax
is-ge
numbe
same-
same-

2.4.3

At pre

binary

higher

in a 'b

be qui

reliab

in tha

for po

'Corn

relatio

such

more

From

extrac

edges

car cl

Ta

and a

Table 2.3: Relations produced by the geometric inference engine

Relation name	Type	Description
angle	orientation	angle between two orientation parameters
coincident-with	proximity	indicates a curve lies in a surface
coplanar-with	containment	indicates 2 planes are coincident
distance-interval	proximity	minimum and maximum distance between two curves
identical	identity	indicates two curves are identical
in-plane-bbox	containment	indicates curve lies in a bounded plane
is-axis-of	identity	indicates line is axis of a surface of revolution
is-generatrix-of	identity	indicates curve is generatrix of a surface of revolution
member-of	identity	indicates curve is part of a composite curve
same-radius	correspondence	indicates two primitives have identical radii
same-type	correspondence	indicates two surface of revolution are the same type

2.4.3 Higher-order Relations

At present, our system computes attributes for single primitives (unary relations), and binary relations between pairs of primitives. It is possible to define tertiary or even higher-order feature types, especially in the case of polyhedral objects. For example, in a 'blocks world' allowing trihedral objects, the tertiary *corner* feature type might be quite valuable at recognition time (if one can insure that corners are accurately and reliably detected). At present, we choose not to extend our geometric inference engine in that direction; however, such relations could easily be added. The *corner* relation for polyhedra could be detected by detecting coincident endpoints of line primitives. 'Corners' also exist for some curved objects. However, the utility of explicit corner relations might be limited due to the difficulties encountered when trying to detect such features. An underlying principle in our object recognition system is that the more reliably a feature can be estimated, the greater the value it has for recognition. From that viewpoint, surfaces are more valuable than edges, because they can be extracted more easily from range data than edges. Corners are less reliable than edges, since small changes in the locations of segmentation boundaries in an image can change the location of a corner feature.

Table 2.3 lists each relation produced by the geometric inference engine, its type, and a brief description.

2.4.4

Earlier

inter-s

versal

makin

approx

edge.

for the

2.4.5

In the

model

are th

depen

object

witho

On

to mo

featur

for a

questi

system

In

viewp

power

we wi

object

the C

inform

Repr

We ta

set of

2.4.4 Calculating Connectivity

Earlier, we highlighted a drawback of IGES as a representation: the lack of explicit inter-surface connectivity information. Connectivity is easily extracted from the universal file generated by GEOMOD (see Section 2.4.5), which contains point lists making up the model surfaces. Our procedure steps through the polygonal surface approximations for each pair of model surfaces extracting points on their common edge. As an additional feature (used during match validation), a 3D bounding box for the edge is calculated and stored along with the connectivity information.

2.4.5 Viewpoint-Dependent Features

In the previous section, we noted that patch areas are added to the node attributes of model surfaces. Why are surface areas important in 3D object recognition, and how are they calculated? Compelling arguments have been made for the use of *viewpoint-dependent* quantities in model-based vision. In fact, some recognition systems classify objects by comparing the sensed scene against 'sample' views stored in a database, without referring to a model-centered description at all [37].

One significant drawback for the use of viewpoint-dependent description is related to model organization. How are 'representative' viewpoints chosen? How are the features calculated from each viewpoint linked? How many viewpoints are 'enough' for a particular object, sensor, or application? Many researchers are attacking these questions; recent work on aspect graphs has produced interesting results, but complete systems have not yet appeared (see Section 2.3.2).

In our work, we have found that an integration of both viewpoint-invariant and viewpoint-dependent features in a model-based 3D object recognition system offers powerful advantages over the use of either type of description by itself. However, we wish to stress that the viewpoint-dependent features play a *subordinate* role in object recognition. The primary source of geometric knowledge about objects is the CAD model; surfaces in the model are *augmented* with the viewpoint-dependent information.

Representative Views: The Uniform View Sphere

We take a practical view of the construction of viewpoint-dependent feature lists. A set of viewpoints on the *view sphere*, which provides a near-uniform sampling of this

finite spa
object is
point in
This ran
to estim

A to
tron (th
and Bro
note the
are give

where

Each
its sid
interi
push
have
the f
obje
the r

Syn

Give
a ra
to t
file
soft
amo

finite space, is selected. Then, for each viewpoint in that set, a range image of the object is generated, along with a synthetic segmentation which associates each surface point in the synthetic range image with the corresponding surface in the CAD model. This range image and segmentation are then used by the feature-extraction routines to estimate the viewpoint-dependent features of interest.

A total of 320 points on the viewsphere are generated from a subdivided icosahedron (the 20-sided regular polyhedron), with its vertices on the unit sphere. Ballard and Brown [3] give an algorithm for constructing a subdivided icosahedron. Let t denote the golden ratio $\frac{1+\sqrt{5}}{2}$. Then the coordinates of the 12 vertices of the icosahedron are given by

$$\begin{array}{cccc} (& 0, & \pm a, & \pm b) \\ (& \pm b, & 0, & \pm a) \\ (& \pm a, & \pm b, & 0) \end{array} ,$$

where

$$a = \sqrt{\frac{t}{\sqrt{5}}}, \qquad (2.2)$$

$$b = \frac{1}{(\sqrt{t\sqrt{5}})}. \qquad (2.3)$$

Each of the triangular faces of the icosahedron is split into 16 subtriangles by dividing its sides into four line segments of identical length and forming triangles. The new interior vertices are not guaranteed to lie on the unit sphere, however; they must be 'pushed out' to the sphere along their direction vector. After the 320 triangular faces have been obtained, a *view direction* for each facet is taken as the vector normal to the facet. Therefore, we obtain 320 viewpoints from which synthetic images of the object are generated. Figure 2.14 shows the icosahedron, the subdivision process, and the resulting 320-sided polyhedron.

Synthetic Range Image and Segmentation

Given a point on the viewsphere (which represents a viewing direction), we synthesize a range image and a corresponding segmentation by using a polyhedral approximation to the object. In addition to an IGES object description, GEOMOD can create a text file known as a 'universal' file; this allows models to be exchanged between GEOMOD software executing on different hardware platforms. The universal file contains a large amount of information about the object, its creation history, and several geometric

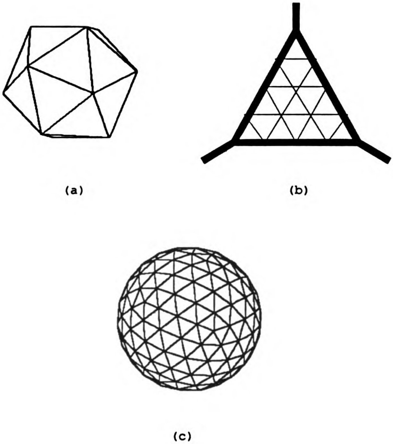

Figure 2.14: Subdivision of icosahedron faces into 16 subfaces. (a) Icosahedron. (b) Subdivision of an icosahedron face into 16 subfaces. (c) Resultant 320-sided polyhedron. The view directions are chosen to lie at the center of the triangular faces.

specifications. Among these specifications is a list of 3D polygons which lie on the object surface. During the design process, this polygonal approximation can be refined to any desired accuracy; this allows curved surfaces to be well approximated for the task of synthetic range image generation. The models we designed for this dissertation had their polyhedral approximations refined so that the maximum deviation from a polygon to its corresponding curved surface was 0.001 inch, comparable to the noise level of our 3D sensor.

We now introduce some notation for the polyhedral approximation produced by GEOMOD. An object $\mathcal{O}$ is a list

$$\mathcal{O} = \{M_1, \ldots, M_n\}, \text{where}$$

each M_i is a *model surface* appearing as a single surface in the IGES file. A common point list

$$\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$$

contains all of the 3D points serving as vertices of the planar facets making up the approximation. Each surface M_i is further decomposed into a list of polygonal facets:

$$M_i = \{f_{i1}, \ldots, f_{in_i}\},$$

where the f_k are the facets and n_i is the number of facets in model surface i. Each facet is defined by its bounding vertices; here, the indices of the points in the point list $\mathcal{P}$ are used:

$$f_j = \{k_{j_1}, \ldots, k_{j_{n_j}}\}.$$

While the notation is somewhat cumbersome, it is a concise way to express the hierarchical structure of the object (the object built from surfaces, each surface built from a number of facets, each facet being built from 3D points). The common point list assures that connectivity information is not lost if the object undergoes a transformation. The polygonal approximation to the object of interest is easily extracted from the universal file produced by GEOMOD. When a synthetic range image is desired, the point list $\mathcal{P}$ is transformed (rotated) according to the desired viewpoint. Then, a process known in the computer graphics literature as 'polygon scan conversion' is applied to each polygon [96]. Scan conversion of a single polygon is accomplished by finding the subrectangle of the image which is occupied by the transformed polygon and sequentially filling in the pixels within this box with the appropriate range information. At the same time, an auxiliary 'label' image is updated with the model urface

range

proxim

sed b

the pi

that r

conve

O

age.

which

Howe

syste

View

From

the

surfa

reco

easi

thei

con

ther

exa

resp

incr

addi

V

addi

based

edge

culate

[6]Su

surface number corresponding to the polygon being scanned. In this way, a synthetic range image and segmentation are generated simultaneously from the polyhedral approximation to the 3D object. Contrast this approach with the ray-casting method used by Hoffman [56]; the ray-casting approach sequentially examines rays through the pixels in the output range image and finds those model surfaces which intersect that ray. As the number of model surfaces (here, facets) increases, polygon scan conversion becomes more computationally efficient than ray-casting.

Our use of the universal file and its polyhedral approximation has one disadvantage. The universal file format is specialized to the GEOMOD software, and the code which makes use of it must be modified for use with a different modeling system. However, a polyhedral approximation to an object is usually available from CAD systems.

Viewpoint-Dependent Features: Visible Area and Simultaneous Visibility

From the synthetic range image and its segmentation, we can produce estimates of the area of visible patches in the image. These quantities (computed for each model surface and view) are used to prune the interpretation tree produced during the recognition phase. In addition, a binary feature defined on pairs of patches is also easily calculated. Two patches are **never simultaneously visible** if, for all views, their areas are never simultaneously positive. This visibility feature also provides constraints on interpretations during the recognition procedure.

Our procedure for calculating area proceeds as follows. For each view, the synthetic segmentation (the label image) is examined. Each 2 × 2 block of pixels is examined. If all four labels for those pixels are identical, the visible area of the corresponding model surface is incremented by the 3D area of the 2 × 2 block. This increment is calculated by breaking the 2 × 2 block into two triangular facets and adding the areas of those triangles together.

While our system only calculates area and visibility features from the CAD models, additional features might also prove useful. Ikeuchi and Kanade [62] have used EGI-based features in their vision system. We could also augment the view-independent edge information extracted from the universal file with visible edge characteristics calculated from the sample views[6]. The selection of the correct 'mix' of view-dependent

[6] Such edge features might not be very useful in occluded scenes, however.

and view-independent features for object description is still an open problem.

2.4.6 Implementation Details

The geometric inferencing system has been implemented on Sun-4 workstations in C and Common LISP. After designing the object using the GEOMOD software, a file containing the IGES information is translated by a C program into LISP S-expressions. Each object is identified by a name (*e.g.* `cube`, `hurdle`, `sundial`). Instances of object types are LISP structures, with corresponding slots and values.

Each primitive in the object description has a unique label in the graph structure. For example, if we have an object labeled `baseball-bat` represented as a single surface of revolution, the axis of the revolved surface would be associated with a structure named `baseball-bat-line`*n*, where n is an integer that distinguishes this line from all other `baseball-bat-line`*m* in the model, $m \neq n$ (actually, n is taken from the position of the entity in the IGES file, and is unique for all entities of any type in the model). The structure contains slots for the parameters of the line (starting and ending points and length). Binary relations affecting an entity are stored in a slot labeled `RELATIONS`. The visible-area features calculated from the viewsphere sampling are placed in a slot labeled `AREA`.

Most computations of the node attributes and binary relations are performed in C. These attributes and relations are assembled into the graph structure within LISP, and connectivity information is constructed. We chose to use C for numerical processing because of its speed advantages over LISP. Once the graph is available, however, LISP allows code which traverses the graph structure to be easily developed. The viewpoint-independent features calculated directly from the IGES file take a few minutes to compute. Generation of synthetic views and calculation of patch areas for each view takes between twenty minutes and one hour per model, depending on the complexity of the polyhedral approximation in the universal file.

2.4.7 Examples

In this section we give examples of relational graphs constructed for two of the objects in our database. Our CAD models were constructed from physical prototypes whose dimensions were measured by hand. In the manufacturing environment, CAD models are often built from designs without reference to physical examples.

Fig

Polyh

Figure
hedra
gives
of ver
the p
turni
was c
Figu
A se
inter
leavi
appe
infor
on, a
whic
flags
obje

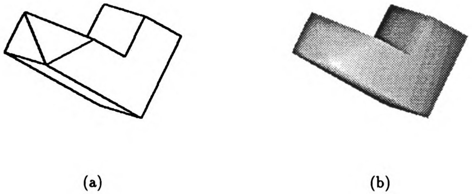

Figure 2.15: Images of a polyhedral object. (a): edge map. (b): range image

Polyhedron

Figure 2.15 shows an edge map and a corresponding synthetic range image of a polyhedral object. This object has twelve vertices (3D points) and nine faces. Table 2.4 gives the coordinates of the vertices and the list of faces, expressed as a sequence of vertices which bound the face when traversed in order. For example, face 1 is the plane bounded by a path through the following vertices: 1, 9, 10, 5, 6, 2, returning to 1. The units of the 3D vertex coordinates are inches. A CAD model was constructed from a prototype of this object using CSG techniques, as shown in Figure 2.16. First, a parallelepiped of dimensions $4.37 \times 3.15 \times 2.87$ was instantiated. A second parallelepiped was subtracted, leaving the L-shaped block. This block was intersected with a bounded halfspace, which removed a corner of the L-shaped object, leaving the block in its final form. A portion of the IGES description of the object appears in Section A.1 of the Appendix. An IGES file contains a header containing information about the software producing the file and the machine it was executed on, along with some global parameters. Following the header is a directory section, which contains pointers to the geometric data for each model entity, along with some flags (we are only interested in the flag indicating the visibility or invisibility of the object). Following the directory is the parameter data section, which contains the

Table 2.4: Vertices and facets for the polyhedral object.

Point	Coordinates
1	(0.00,0.00,0.00)
2	(4.37,0.00,0.00)
3	(4.37,0.00,-2.87)
4	(0.00,0.00,-2.87)
5	(1.65,1.57,0.00)
6	(4.37,1.57,0.00)
7	(3.00,1.57,-2.87)
8	(1.65,1.57,-2.87)
9	(0.00,3.15,0.00)
10	(1.65,3.15,0.00)
11	(1.65,3.15,-2.87)
12	(0.00,3.15,-2.87)

(a) Model vertices

Facet	Point List
1	1 9 10 5 6 2
2	2 6 3
3	6 7 3
4	5 8 7 6
5	5 10 11 8
6	9 12 11 10
7	9 1 4 12
8	1 2 3 4
9	11 12 4 3 7 8

(b) Point lists for model facets

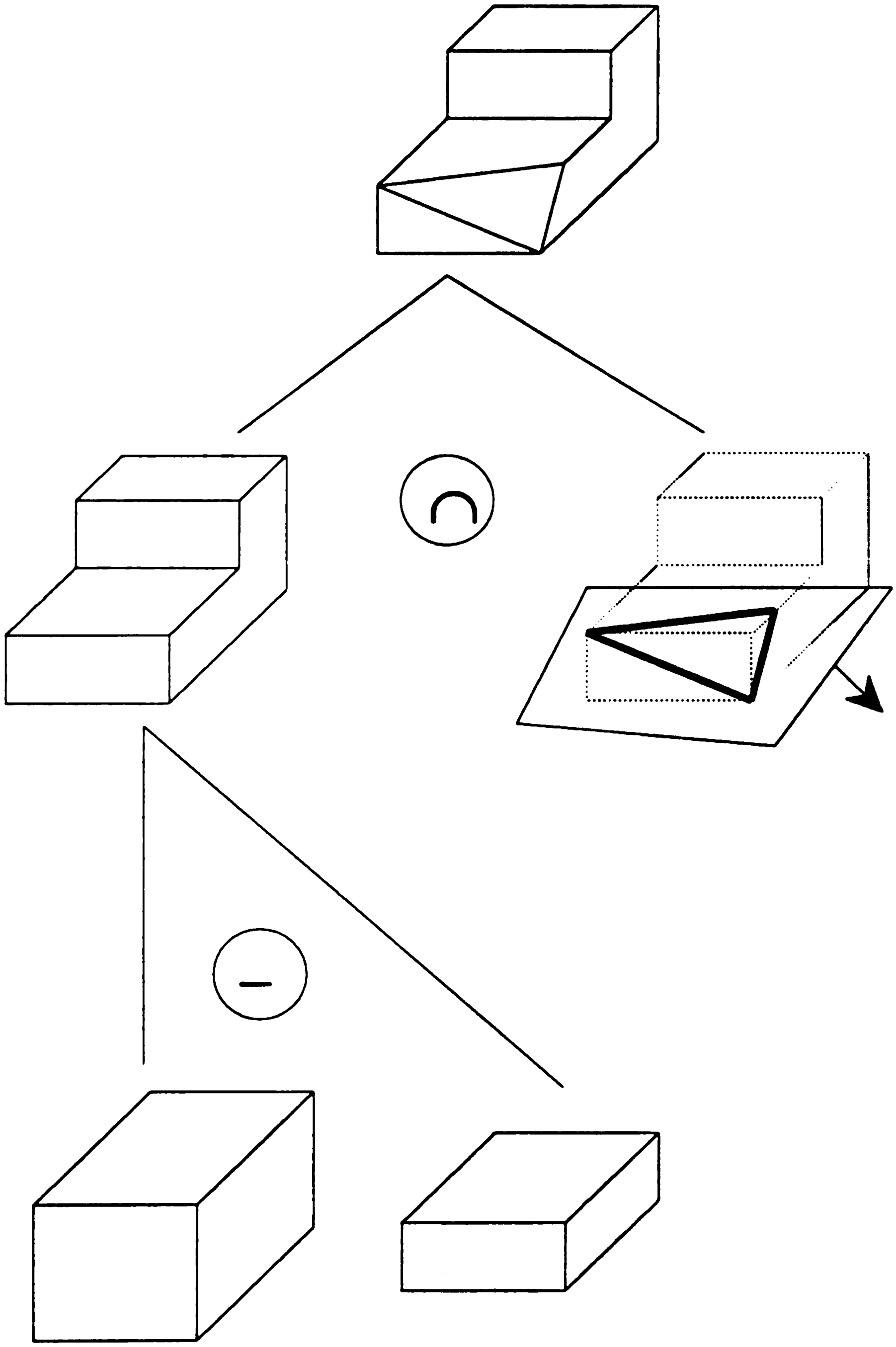

Figure 2.16: The CSG model for the polyhedron.

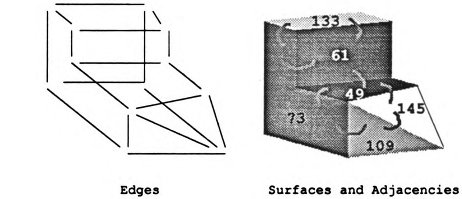

Figure 2.17: A portion of the relational graph for the polyhedron

geometric parameters for each entity. The C language part of the inference engine produces approximately 2000 lines of LISP code; selected lines appear in Section A.1 of the Appendix. About 400 of the S-expressions are descriptions of nodes in the relational graph and 'housekeeping' statements allowing the graph to be traversed easily. The remainder of the S-expressions are binary relations including connectivity and visible-area quantities. A graphical rendition of the relational graph appears in Figure 2.17. Only a few of the relations are drawn on the graph.

A curved object

Figure 2.18, parts (a) and (b), show hidden-line and synthetic range images of a curved part with rotational symmetry. The IGES file for this part contains four cylinders, five planes, ten circular arcs, and a few additional invisible primitives. A few of the nodes and edges in the relational graph are depicted in Figure 2.19.

2.5 Object Model Database

Twenty solid models were constructed for use with the BONSAI recognition system. All were created using the GEOMOD software; relational graphs were constructed from the IGES and universal file descriptions using the methods described above. The models were constructed using the CSG approach, using measurements taken on physical examples. In the brief descriptions below, we will note that several models

Figure 2.18: Hidden-line and synthetic range images of a rotationally-symmetric part.

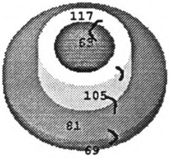

Figure 2.19: A portion of the relational graph generated for the rotationally-symmetric part.

are sim

usually

deep co

less co

were us

concen

primiti

Syn

ing col

We n

1. A

2.

3.

4.

5.

[7] T

comp

state

values

are simplified representations of the actual physical objects. These simplifications (usually the covering of object holes) were made primarily because surfaces within deep concavities on the object are unlikely to be detected by our range sensor, much less correctly classified. If a different range sensing strategy (such as FM radar) were used, we might wish to increase the level of detail of the models. For now, we concentrate only on representing the gross shape of the object using three types of primitive surfaces.

Synthetic range images of the twenty objects appear in Figure 2.20. A corresponding collage of 'pseudo-intensity' images for the same objects appears in Figure 2.21[7]. We now briefly describe the objects used in our experiments.

1. **ADAPTER**

   This object is a fitting used to change the diameter of plastic piping. We covered the ends of the object with paper for the purposes of design and image acquisition; therefore, the model and our images do not contain the hole through the center. Shadowing effects would make reliable extraction of the parameters of the inner curved surfaces difficult in any case. The solid model consists of three planes (all parallel), and two cylinders of slightly different radii (1.125 and 1.375 inches).

2. **AGPART2**

   This rotationally-symmetric object was obtained from a machine shop on the Michigan State University campus. Its solid model consists of five cylinders and five planes.

3. **BALL**

   This object is a child's ball made of rubber, with a radius of 3.5 inches.

4. **BALL2**

   This object is a spherical globe from a lighting fixture; its radius is 3 inches.

5. **BIGWYE**

   This object is a plastic plumbing part known in the trade as a 'wye' because

[7] The intensity value at each pixel in the pseudo-intensity collage is proportional to the z-component of the surface normal at the corresponding location in the range image. Fan *et al.* [37] state that artificial shading of this type makes better use of the dynamic range of displayable gray values.

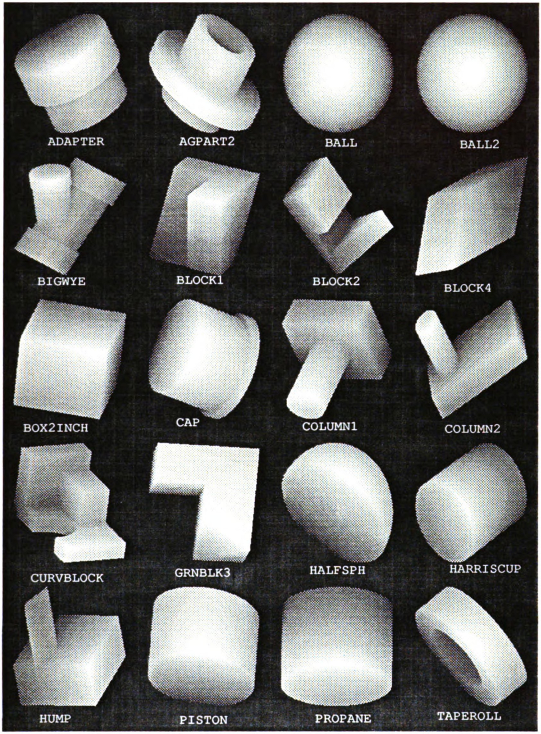

Figure 2.20: Synthetic range images of twenty objects.

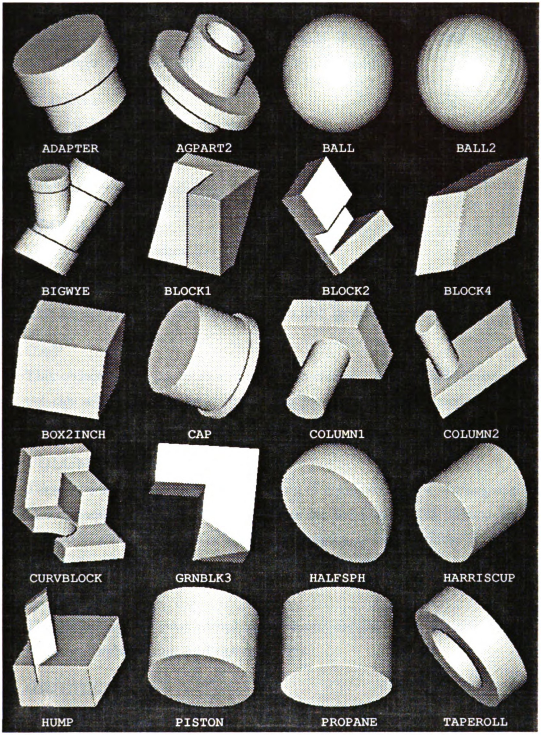

Figure 2.21: Pseudo-intensity images of twenty objects.

it is used to join two pipes at an angle. The join angle is 45 degrees, and the arms of the object have different outer radii. As with the ADAPTER object, we judged that the parameters of the inner surfaces would be difficult to obtain because of shadowing artifacts, so the ends of the pipe sections were covered with paper and specified as planes in the solid model.

6. **BLOCK1**

   This polyhedron was constructed from wood. It has eight planar faces.

7. **BLOCK2**

   This polyhedron contains ten faces.

8. **BLOCK4**

   This convex polyhedron has six faces.

9. **BOX2INCH**

   This object is a cube with a side length of 2 inches.

10. **CAP**

    This object is the cap from a can of glass cleaner. We covered the open end of the cap with paper to avoid shadowing problems with parameter estimates of the inner surfaces.

11. **COLUMN1 and COLUMN2**

    These two objects are similar, consisting of a cylinder and a rectangular block. The cylinder is centered in a square face for the COLUMN1 object. The rectangular block is larger in the COLUMN2 object and the cylinder is affixed near to one side of a planar face, rather than being centered.

12. **CURVBLOCK**

    This object is the most complicated in our database. It consists of thirteen surfaces (twelve planes and one cylinder).

13. **GRNBLK3**

    This object is a nine-sided polyhedron.

14. **HALFSPH**

    This object is a hemisphere of radius 1.25.

15. **HARRISCUP**

    This model was constructed from a coffee cup. Because it cannot be represented exactly as a planar, spherical, or cylindrical primitive (and the segmentation procedure would usually label it unclassifiable anyway), we did not attempt to model the handle. We did model the inside of the cup, however.

16. **HUMP**

    This object consists of a half-cylinder cutout affixed to a rectangular block.

17. **PISTON**

    This object is a large cylindrical piece of metal. The object from which the model was constructed has a complicated crankshaft attachment that we did not attempt to model.

18. **PROPANE**

    This object is a cylindrical propane tank for use with a small outdoor grill. We did not model the threaded fitting used to attach the tank to the grill.

19. **TAPEROLL**

    This object (a roll of masking tape) is a cylindrical ring of inner radius 1.5, outer radius 2.5, and height 1.5 inches.

2.6 Summary

In this chapter, we have described a geometric inference engine which produces a relational graph representation of a 3D object from a geometric model specified using the IGES design standard along with a polyhedral approximation. While the GEOMOD solid modeler was used in our work, any 3D CAD system which generates object descriptions using the 'analytic' IGES form could be used instead. In addition to the process of translation from one data format to another, the inference engine extracts higher-level information from the CAD model and stores it explicitly in the new data structure. The higher-level features will allow the search space explored during the object recognition stage to be pruned early. The view-independent features extracted from the IGES model description are supplemented with view-dependent features (areas and simultaneous visibility lists) for each model surface taken from 320 views of the object.

The contribution of the work reported in this chapter is twofold.

- The model-building step in machine vision has traditionally assumed one of two forms: construction by hand, or construction by learning from images (often requiring human interaction). The availability of object geometry in CAD databases motivated our construction of a model-building system which constructs relational graphs *automatically*, without human intervention.

- Secondly, we desire to separate a strategy for recognition of 3D objects into on-line and off-line phases. The off-line procedure described here allows the recognition process to become a traversal of an existing database rather than the construction of such a database at recognition time.

Ch

3D

3.1

The r
know
dition
sensi
ditio
than

T
been
imp
with
In p
con
tech
pro
the
the

ser
sev
(u
na
be
si
fo
e

Chapter 3

3D Sensing

3.1 Introduction

The model-based vision system described in this dissertation uses *range images* (also known as *depth maps*) as its input. Range images enjoy many advantages over traditional intensity maps for computer vision tasks, but hardware (and software) for sensing depth is often more complicated and expensive than intensity sensors. In addition, depth sensors typically take longer (seconds to minutes) to produce an image than intensity sensors.

The specification of sensing methods (the actual physics of the sensor) has often been *decoupled* from the vision task by researchers; their philosophy regarding sensing implies that as long as the format of the image input to the processing step agrees with the output of the sensor, any sensor providing that sort of data may be used. In practice, the interface between sensing and processing is not a sharp boundary; consider the extraction of range maps from intensity imagery, using shape-from-X techniques. In this situation, we would still be tempted to call the input to the processing stage a range image, and lump the processing necessary to derive it from the intensity information into the sensing process. Some recent work has addressed the impact of sensor specifications on visual tasks [54, 62].

While many range sensors provide data in a R_{ij} (grid or raster) format, R representing range and (i, j) representing image plane coordinates as above, other range sensors simply provide lists of Cartesian coordinates $(x_i, y_i, z_i), i = 1, ..., n$, for some (usually large) value of n. In the first format, connectivity in image plane coordinates is explicit. When the image is presented as a list of 3D points, connectivity between neighboring points can be established by placing the data in a k-D tree or similar data structure [42]. Clearly, R_{ij} form can easily be transformed into Cartesian form, but the converse transformation is not possible in general. Most vision research employing range data has employed the R_{ij} format for synthetic and real imagery.

The problem domain often dictates the design parameters of the range sensor [68].

Table 3.1: Range sensor requirements for different application areas [68].

Application Area	Description	Spatial Resolution	Data Rate (pixels/sec)
Integrity and Placement	Detect missing or misaligned components	$10^{-2} - 10^{-1}$ inch	$10^{5} - 10^{6}$
Metrology	High-precision measurement	$10^{-6} - 10^{-4}$ inch	$10 - 10^{5}$
Surface Inspection	Exhaustive scan for flaws and assembly defects	$10^{-4} - 10^{-2}$ inch	$10^{4} - 10^{6}$
Modeling	Construct a solid model from multiple images	$10^{-4} - 10^{-2}$ inch	$10^{4} - 10^{6}$
Recognition	Recognize and locate objects in an image	$10^{-4} - 10^{-2}$ inch	$10^{4} - 10^{6}$
Navigation	Obtain scene data for autonomous vehicle	1 – 10 inches	$10^{4} - 10^{6}$

Table 3.1 lists a few application areas and the constraints in terms of spatial resolution and data bandwidth.

3.2 Survey: Range-Finding Techniques

In this chapter we briefly describe most of the techniques currently being used to acquire range data from a scene. This discussion borrows heavily from two excellent surveys, Besl [7] and Nitzan [87], and is meant to give a flavor for different sensor types rather than a detailed description of methods; for details, the reader is referred to these surveys, or Jarvis' earlier work [69].

In attempts to group range sensors into classes, it is often informative to describe them in terms of the following properties.

- *Active vs. passive*: an active sensor illuminates its environment. A passive sensor does not.

- *Monocular vs. multinocular*: A monocular sensor produces one image. A multinocular image produces more than one. Binocular stereo sensors process two intensity images in order to obtain depth. Some laser radar systems produce registered range and intensity images. Multiple images can provide complementary information, but the question of integration of visual information has only recently been addressed by researchers.

ot

an

t p

me

ak

.2

ad

ate

he

our

I

(fi

he

ase

o a

en

s l

pro

of

for

ra

be

su

th

ra

de

to

a

res

a

99

Not all sensors provide absolute depth information; some only produce the relative change in depth between distinguished scene points. Photometric stereo sensors do not produce range data at all, but instead produce an image of surface normals. In some applications, surface orientation may be the most important feature to obtain, making such sensors preferable to other types.

3.2.1 Radar Sensors

Radar-based range sensors are often labeled *active* devices because the sensor illuminates the object with a light source (often a laser), and range is calculated based on the reflected energy. Most range sensors of this type employ lasers as the illumination source, yielding the term *laser range finder.*

Perhaps the most intuitive use of radar techniques for range sensing is simple time-of-flight measurement: since electromagnetic radiation travels at a constant speed, the distance to a point on an object's surface can be determined by directing a pulse of laser light to that point and measuring the time interval needed for the reflected pulse to arrive back at the sensor. This sensing method is quite feasible for long-distance sensing, such as in laser radar range finders for target acquisition, where the sensor is located several hundred meters from the objects to be sensed. Such sensors often produce both range and intensity images (the intensity is derived from the amplitude of the returning laser light pulse); Jain *et al.* have investigated sensor fusion issues for automated target recognition using such data [65]. Small-standoff time-of-flight range finders have been constructed but the precision of the optical equipment must be high as the time intervals to be measured are extremely short [119].

An alternative to measuring pulse flight time is to continuously illuminate the surface point with an amplitude-modulated laser signal. The phase difference between the incident and reflected energy is proportional to the range value sought. AM range sensors' most significant problem is the *ambiguity interval*: the range value is determined only up to a value within an interval whose length is inversely proportional to the frequency of the amplitude modulation. In other words, for a sensor returning an N-bit range value, the AM frequency may be increased, providing greater range resolution in a smaller ambiguity interval, or it may be decreased, enlarging the ambiguity interval but coarsening the range value quantization correspondingly [81, 99].

In
th
he
s.

.2.

ias
or
on
g-
ore
oin
ec
llov
a'b
req
si
A
ial
te
r (
r s
sec
n a

3.2

Pas
lep
kr
o t
D
oi
rac

Instead of modulating the amplitude of the laser signal, the frequency of the light radiation can be linearly swept between two values in a certain time interval (the sweep period), and range can be estimated using coherent heterodyne detection. Besl [7] reports on a few sensors of this type.

3.2.2 Triangulation-Based Sensing

Triangulation sensors are very inexpensive to build, requiring none of the precision optics and/or electronics of the laser range finders described above. A single monochrome camera and a light projector are sufficient hardware to construct a triangulation sensor based on structured light [60]. The software to derive depth is often more complex, however. The basic idea of triangulation sensing is straightforward: a point (x, y, z) in the 3D world images to a point (u, v) in the image plane. *A priori* specification of the geometric relationship between the light source and the camera allows us to relate a 3D point (illuminated by the light source) and its 2D projection (a 'bright' pixel in the intensity image produced by the camera) by similar triangles. Frequently, the geometric relationship is obtained through a calibration procedure. A simple diagram of a triangulation sensor appears in Figure 3.1.

A variety of methods for triangulation sensing have been developed into commercial and 'home-brewed' sensors. Some sensors project an individual point of light on the object and move the point over the object surface to obtain image data (either R_{ij} or (x, y, z) format). Other sensors use projected lines of light, grids, coded stripes, or synthetic texture to obtain multiple range points from a single image. The sensor used in our research uses the triangulation principle to derive depth; details are given in a later section of this chapter.

3.2.3 Stereo Techniques

Passive stereo techniques employ constraints between two or more cameras to derive depth. In *binocular stereo*, the image plane coordinates of two cameras are related by a known transformation. Identification of points in the two images which correspond to the same 3D point in the scene allows us to calculate the coordinates of that 3D point. The difficult problems in stereo vision are the selection of 'distinguished' points in the two images and the establishment of correspondences. Typically, high-gradient (*i.e.*, edge) points are chosen as candidate points for correspondence, and

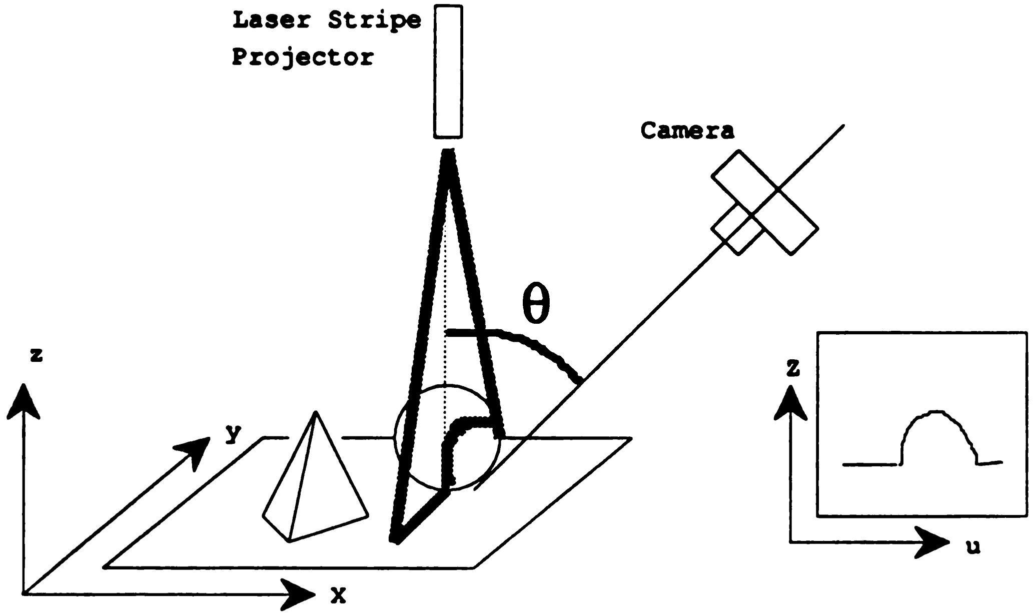

Figure 3.1: Triangulation-based range sensing

heuristic search procedures attempt to form the correspondences [112]. Extensions to three cameras (*trinocular* stereo) have recently been proposed [50, 106]. Stereo vision systems have been used in outdoor scenes for navigation [105].

3.2.4 Shape-From-X

Can we recover the shape of a 3D object from a single intensity image? Researchers have approached this problem from many directions, producing techniques known collectively as 'shape-from-X' methods. Horn [59] develops the shape-from-shading method by exploiting constraints on the *reflectance map* (which relates surface orientation to image brightness). *Photometric stereo* [59] is another technique, employing multiple light sources and monocular intensity images in order to recover 3D shape. The photometric stereo method does not recover absolute depth, however. While shape-from-shading recovers surface orientation directly, it can be used to estimate range values as well. Vaillant and Faugeras [115] developed a system to estimate the depth map along occluding image contours (shape-from-contour). Nalwa [84] inferred surfaces of revolution from bilaterally symmetric image contours. Blostein and

Ahuja [13] extract surface shape by estimating the perspective distortion of texture elements extracted automatically from the unknown scene (*i.e.* the texture primitives themselves are selected from the image, not specified *a priori*).

3.2.5 Other Approaches

A novel approach to range estimation is to examine a pattern of Moire fringes projected onto the scene entities [7]. A Moire pattern is produced by passing light through two gratings which 'interfere' with one another through a slight angular offset. Often the two gratings are identical and contain a pattern of equal-width opaque and transparent stripes. When two of these gratings are overlaid with different orientations, a pattern of interference fringes is produced. Moire fringe images are often obtained by projecting light through one grating and placing the other grating in front of the camera lens. Relative depth between centers of adjacent interference fringes is fairly easily computed, but the absolute depth is harder to find, requiring some 'ground truth' about the true location of a few points in the scene. Many additional variations on the Moire theme have been described in the literature, but the difficulty in extracting absolute depth has restricted the application of these sensors.

Fairly flat objects can be measured using holographic interferometry in a process similar to the Moire technique above. Instead of two gratings producing an interference pattern on the scene, two lasers (with identical polarization) illuminate the object, also producing an interference pattern on the surface. The pattern can be analyzed much as a Moire pattern to yield relative depth between adjacent bright stripes in the pattern.

Depth from focus techniques exploit the law of thin lenses to obtain range [73]. Given a lens of focal length f, a point z units in front of the lens, and a distance w behind the lens upon which the point at z is focused,

$$\frac{1}{w} + \frac{1}{z} = \frac{1}{f}.$$

Assuming that the thin-lens model holds, range can be determined for an illuminated scene point as long as it can be determined *when* the point is in focus on the image plane. Depth-from-focus seems to have high potential for the future in vision research; several such sensors have been constructed [90, 111].

Fresnel diffraction illuminates the scene with coherent light passed through a diffraction grating. The physics of coherent light diffraction state that the light

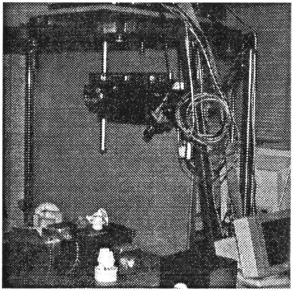

Figure 3.2: The range sensing worktable

reproduces the grating pattern at periodic intervals. Within these intervals range can be estimated from the local contrast in the image, as the 'focused-ness' of the diffraction grid image depends upon relative position within the interval. This has formed the basis for a few range sensors.

3.3 Technical Arts 100X scanner

This section describes the Technical Arts 100X Scanner [113], a structured-light range finder using a laser for active illumination. A photograph of our range sensing workbench appears in Figure 3.2.

Figure 3.1 illustrates the general principles of the sensor and the sensor coordinate system[1]. A tube creates the coherent laser radiation with a power level of 5 milliwatts. The beam travels horizontally from the tube's aperture, reflects off of a mirror and strikes an oscillating mirror. The oscillating mirror creates a vertical triangular plane of light which falls on the objects on the work table. The sensing footprint is about 12 × 12 square inches. Instead of using a mirror to move the plane of light, two motorized stages are used to position the object under the plane of light. One stage

[1] The origin of the sensor coordinate system is fixed during the calibration procedure.

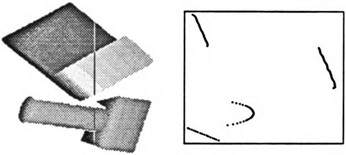

Figure 3.3: Range values for a single laser stripe.

translates in the x direction, and the other moves the object in the y direction. At present, we do not perform any y translations of the object during a scan.

A CCD camera (with a 240 × 240 sampling grid) is positioned so that its optical axis intersects the plane of laser light. The angle between the camera axis and the z direction is 45 degrees. The image and sensor coordinate systems are aligned so that position along the stripe (the y coordinate) is proportional to the row number in the image plane, and height above the tabletop (the z coordinate) is proportional to the column number in the image produced by the CCD camera. For each constant-x range profile, the corresponding profile in the image plane is an irregular, broken contour. Calibration of the position of bright spots in the profile against their known positions in 3D allows the y and z coordinates to be determined. This process is applied to each row of the CCD camera image, yielding up to 240 depth values along a slice. After the slice is processed, the x stage can be translated and the process repeated. Note that the distance between slices need not be the same as the distance between adjacent rows of a slice. In our work, we adjust the number of slices and the spacing between slices in order to keep the aspect ratio between 0.5 and 2.0.

Figure 3.3 shows a range image with a highlighted column corresponding to a single scan, and the 3D points extracted during that scan.

The output of the 100X scanner is a grid-format range image

$$\mathcal{R} = \{\mathbf{r}_{ij} = [x_{ij}, y_{ij}, z_{ij}] | x_{ij} = k_x j, y_{ij} = k_y i, 1 \leq i \leq m, 1 \leq j \leq n\}.$$

Note that we know the k_x and k_y values, so the 3D coordinates of each image point are known. We typically calibrate the sensor to produce measurements in inches (although it could easily be calibrated to operate in metric units as well). To calibrate the sensor, an object called a 'calibration gauge' is placed under the laser stripe, as

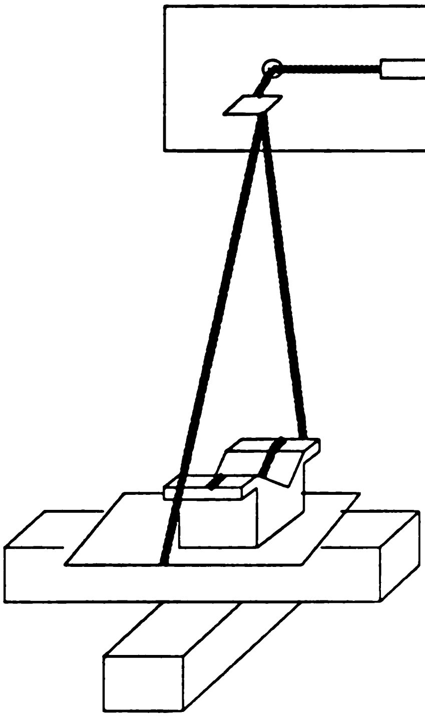

Figure 3.4: Calibration of the 100X scanner.

shown in Figure 3.4. The central V-shaped portion of the gauge is 3.5 inches deep and 7.0 inches across. Repeated scans of this object are acquired, and the pitch and roll of the CCD camera are adjusted after each scan, until the calibration error falls below thresholds supplied by the manufacturer. If desired, the calibration can be improved by placing an additional object (of known dimensions) under the laser stripe, and requiring the sensor to produce the correct range estimate on the additional object's surface.

The y and z coordinates of the range pixels are quantized to 0.001 inch. While some noise is present in the z coordinate, our estimates show that the noise standard deviation is less than 0.01 inch. The shadowing effects mentioned earlier (as well as specular reflections, surface markings, or any other artifact which prevents the CCD camera from seeing a portion of the stripe)[2] yield pixels in the image where the range data is invalid. For convenience, we can view this as a *flag image*

$$\mathcal{F} = \{f_{ij}\}$$

[2] The red laser light will be absorbed by a green surface. Highly specular (mirrorlike) surfaces will usually not reflect much of the incident light back to the CCD camera.

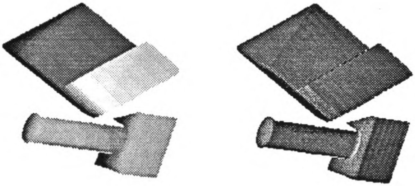

Figure 3.5: Range and intensity images from the 100X scanner.

whose value at a given location is TRUE if the pixel is valid and FALSE if the pixel's 3D coordinates should be ignored. In addition to pixels which *would* lie on the object surface if they were visible, pixels lying on the 'background' of the image are also flagged as FALSE, because the range sensing platform is painted black, and it absorbs the incident laser radiation.

In addition to the range and flag images, the 100X scanner produces an intensity image, which measures the amplitude (brightness) of the visible laser stripe at each range pixel. This image is registered with the range data, and has invalid pixels at the same image-plane locations as the range image $\mathcal{R}$. We note that the intensity image is measuring 'active intensity', rather than the photometric behavior of the surface under ambient room lighting. Figure 3.5 shows range and intensity images for the objects shown in Figure 3.3. The intensity image is brightest at concavities in the scene, where more of the laser's energy is concentrated. The intensity information may prove useful in image segmentation strategies employing fusion of the two images; we are using only the range data in this thesis.

Like intensity images, range images contain *edges*; they form the boundaries between smooth surfaces of an object, or the boundary between two objects, or the boundary between the object and the 'background.' Like most researchers, we identify two types of edges in range imagery. A depth discontinuity is labeled a *jump edge*. Jump edges are *not* intrinsic properties of the object being sensed, but instead depend on the viewpoint. *Crease edges* are caused by a discontinuity in the surface

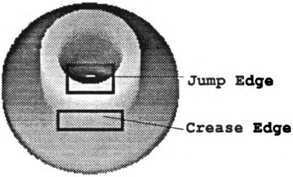

Figure 3.6: Crease and jump edges.

normal of the object, and are therefore intrinsic. These two edge types are illustrated in Figure 3.6. In Chapter 4 we will survey a few techniques for extracting these edges from range imagery. It should be noted that higher-order edges do exist; for example, a *curvature* edge is present where the second derivative is discontinuous. However, difficulties in reliable extraction of second-order properties from range data have limited the use of higher-order edges in scene analysis.

Figure 3.7 shows real range images of the twenty objects in our CAD model database. The models are described in Section 2.5. Corresponding pseudo-intensity images appear in Figure 3.8. Synthetic range and pseudo-intensity images of the models appeared in Figures 2.20 and 2.21, respectively. Shadowing and occasional 'drop-outs' (isolated missing pixels) are visible in the real images.

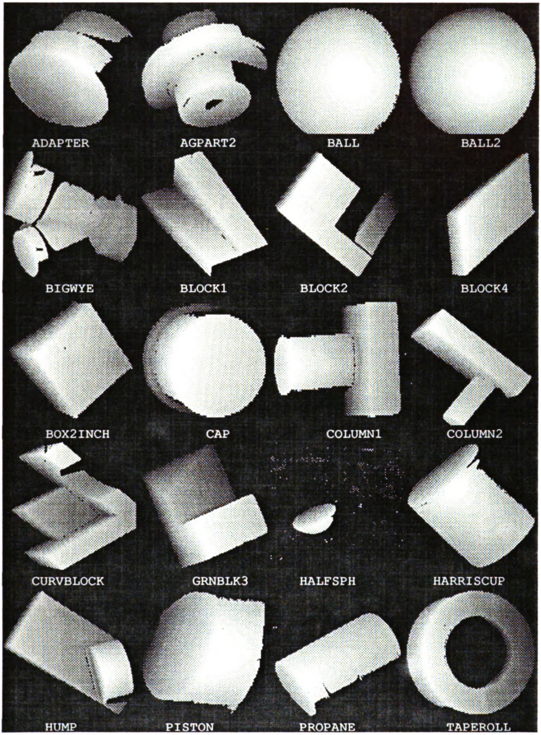

Figure 3.7: Real range images of twenty database objects.

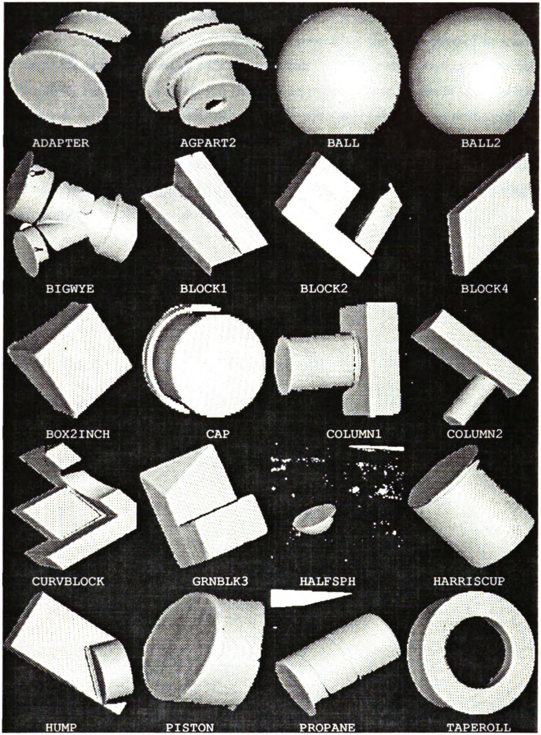

Figure 3.8: Pseudo-intensity images of twenty database objects.

3.4 Summary

This chapter has explored methodologies for 3D sensing, concentrating on active techniques (radar and structured light). We presented the principles of operation of the 100X 3D scanner used in this dissertation. While many existing range sensors are too slow, too expensive, or too difficult to manufacture to make them attractive in industrial environments, the technology is constantly improving and applications well-suited to range data abound [68]. Improvements in sensing technology are needed to make range sensors more attractive to manufacturing; in addition, strategies for low-level and high-level processing of range data are also vital if 3D sensing is to assume a more prominent role in industrial inspection.

Chapter 4

Segmentation and Surface Classification

In this chapter we describe procedures which are applied to range images to extract surfaces and edges. Two processes are involved. In the *segmentation* process, the pixels of the range image are grouped into regions called *segments*. Ideally, we wish to have one image segment for each connected portion of a smooth (at least C^1 continuous) surface on the object. We will occasionally use the term *patch* as a synonym for 'segment'. Segmentation is accomplished by clustering of image pixel coordinates and associated surface normals, followed by cleaning and merging procedures. In the *classification* process, we attempt to characterize the object surface producing an image segment. We currently classify patches as planar, spherical, or cylindrical. These classifications are based on linear and nonlinear fitting techniques and curvature estimators.

The chapter begins with a survey of previous research on range image segmentation. Then, the procedures we employ in our research are explained and demonstrated on two sample images.

4.1 Survey

Range image segmentation and classification are popular topics of research. This section surveys some of the recent directions in the area.

4.1.1 Superquadric Classification.

In Chapter 2, we introduced the superquadric volumetric primitives. Solina and Bajcsy [103], Ferrie *et al.* [38], Gupta *et al.* [48], Allen and Michaelman [2], and Boult and Gross [17] have all addressed the problems in superquadric fitting. Essentially, the problem is a straightforward application of nonlinear optimization. The goal is to fit a single (possibly deformed) superquadric model to a set of n 3D points

$\{(x_i, y_i, z_i) \mid 1 \leq i \leq n\}$. A nondeformed superquadric in general position has eleven parameters: two shape parameters (ϵ_1, ϵ_2), a scale factor for each axis (a_1, a_2, a_3), a rigid rotation expressed as three Euler angles (ϕ, θ, ψ), and a translation vector $(\Delta x, \Delta y, \Delta z)$. If global deformations are incorporated into the fitting process, the number of parameters will be larger.

The Levenberg-Marquardt nonlinear optimization technique [101] seems to be the method of choice for performing the fitting. The superquadric model is expressed as an implicit function of (x, y, z) coordinates and the model parameters, defined to have the value zero when a point on a given superquadric is plugged into the function when the correct parameters are given. Like most nonlinear optimization strategies, Levenberg-Marquardt requires an initial estimate of the model parameters. Much of the recent research cited above has dealt with the merits of various objective functions, and methods for evaluating the quality of a superquadric fit. Often, the range data is assumed to be presegmented; to our knowledge, only Ferrie *et al.* have placed model fitting into the broader context of segmentation.

4.1.2 Curvature-Based Segmentation

Segmentation procedures based on qualitative interpretations of surface curvatures have been used extensively over the last several years [10, 36, 100, 117, 122]. Our detailed discussion of curvature extraction appears later in this chapter. For the moment, we will give an intuitive description. *Principal curvatures* are intrinsic quantities defined at each point $\mathbf{p}$ on a surface. There are two directions tangent to the surface along which the surface curves the most or least. On nondegenerate surfaces, these directions are mutually orthogonal; they are known as *principal directions*, and the corresponding curvatures are called *principal curvatures*. On a cylindrical surface, for example, one principal curvature is zero, and the corresponding direction is parallel to the cylinder's axis. The other principal curvature direction is perpendicular to the axis, and the curvature value is the inverse of the cylinder's radius. The signs of these curvatures depend on our choice of coordinates. A negative principal curvature implies that the surface is curving 'downward' with respect to the coordinate frame; conversely, a positive curvature value means that the surface is curving 'upward' in the associated direction.

Most qualitative segmentation schemes use mean and Gaussian curvatures rather

Table 4.1: Segmentation from signs of mean (H) and Gaussian (K) curvatures (from [10]).

	$H < 0$	H=0	$H > 0$
$K < 0$	saddle ridge	minimal surface	saddle valley
K=0	ridge	plane	valley
$K > 0$	peak	*impossible*	pit

than the two principal curvatures. Mean curvature is the average of the two principal values, and Gaussian curvature is their product. Qualitative segmentation can be performed using the signs of mean and Gaussian curvature. Table 4.1 shows the 8 valid classifications for a point based on the two signs.

Most strategies segment an image by classifying each pixel as one of the eight types. Connected regions of the same type are identified as primitive image surfaces. Fan *et al.* [36] use an edge-based strategy. Directional curvature estimates taken from four directions are combined to obtain principal curvatures and directions. Extrema and zero-crossings in principal curvature locate candidate edge points, which are then grouped into closed contours, producing an image segmentation.

4.1.3 Edge Detection in Range Imagery

In addition to the edge-based segmentation scheme of Fan *et al.* [36] (which used edge pixels suggested by extrema or zero-crossings in curvature), other general edge-finders suitable for range imagery have been proposed. The two edge types most often considered in range imagery are the jump and crease edges. A jump edge occurs at a point of range discontinuity, and jump edge pixels can be detected using any of the operators developed for edge detection in intensity imagery, such as the Sobel or Prewitt operators [3]. A crease edge occurs where the surface normal is discontinuous, and techniques from intensity image processing are not directly applicable. Inokuchi *et al.* [63] examine a ring of pixels centered at the image point of interest and classify the point as a jump edge, convex (roof) edge, concave (valley) edge, or non-edge point, by examining the Fourier spectrum of the depth values in that ring (the ring of depth values is viewed as a periodic 1D signal). Their detector was not tested with curved surfaces, however, and even with FFT hardware, the edge-detection process can be time-consuming, since the Fourier transform must be computed for every pixel in the image.

Mitiche and Aggarwal [80] developed a statistical test for crease edges. At a range pixel of interest, several partitionings of its neighborhood are considered. A partitioning splits the neighborhood into half-neighborhoods, and a planar surface is fit to each 'side.' Under appropriate assumptions, the difference in slope between the two planes in a 'good' partitioning serves as a test statistic. Thresholds on the statistic produce a decision procedure which can be applied at each range pixel, producing an edge or no-edge labeling of the image.

The primary drawback of edge-based segmentation is inevitable fragmentation of the edges. If the edges do not form a closed network, they must be linked together using heuristic techniques. For this reason, edge-based range image segmentation has not received as much attention from researchers as region-based approaches. Nadabar and Jain [83] have combined our local feature clustering approach to range image segmentation with an edge-based segmentation strategy using a Markov random field model for the edge process, with good results.

4.2 Initial segmentation: Local feature clustering

A functional block diagram of the range image segmentation and region classification process appears in Figure 4.1. Our technique for obtaining a preliminary segmentation was used by Hoffman and Jain [57] to segment images obtained from an AM radar range sensor. Surface normals are estimated at each pixel where the range value is valid (see Section 3.3) using principal components analysis; three components of the normal vector (nx, ny, nz), together with the (x, y, z) position of the point, produces a six-dimensional vector at each pixel in the image. A sampling of these vectors is input to a squared-error clustering procedure. The output of the clustering step is a preliminary labeling of image regions. The clustering technique usually oversegments the input image (*i.e.*, it produces more distinct image patches than there are smooth surfaces in the scene); this is remedied through application of a region merging operation. This type of procedure is often referred to in the literature as *split-and-merge* segmentation.

After the preliminary segmentation, a surface classification is obtained for each connected surface patch. Planar faces are extracted first, using an error measure calculated from a principal components planar fit to the 3D points in the patch. Nonplanar surfaces are further subclassified using estimated surface curvatures. At each surface point, we estimate minimum and maximum curvature and their associated directions. The surface is then classified as being spherical, cylindrical, or 'unknown' using curvature values and directions. For each type of surface, the appropriate geometric parameters (*e.g.* radius, orientation) are estimated. If adjacent image patches are found to be of the same surface type with similar parameters, they are merged at this time. Finally, the estimated information for each patch is represented as a number of LISP S-expressions and serves as input to the recognition system.

The following notation will prove useful throughout the chapter.

- The $m \times n$ range image produced by the 100X scanner is denoted as $\mathcal{R}$:

  $$\mathcal{R} = \{\mathbf{r}_{ij} = [x_{ij}\ y_{ij}\ z_{ij}] \mid 1 \leq i \leq m, 1 \leq j \leq n\}.$$

  Note that the pixel's coordinates form a row vector. We will refer to the elements $\mathbf{r}_{ij}$ image as *pixels* (or *range pixels*); it is important to remember that a range pixel has three coordinates. Henceforth, we will not show the bounds on the row and column indices.

- In Chapter 3, we noted that not all of the mn pixels in the image contain valid data, and defined a Boolean *flag image*

  $$\mathcal{F} = \{f_{ij}\},$$

  in which a TRUE value for f_{ij} means that the corresponding range pixel $\mathbf{r}_{ij}$ contains a valid 3D measurement, and a FALSE value indicates that the pixel should be ignored in further processing.

- Surface normals are estimated at each pixel $\mathbf{r}_{ij}$ which is itself valid and has at least two valid neighboring pixels in its 5×5 neighborhood. This produces a *normal image*

  $$\mathcal{N} = \{\mathbf{n}_{ij} = [nx_{ij}\ ny_{ij}\ nz_{ij}]\},$$

  where nx_{ij}, ny_{ij}, and nz_{ij} denote the components of the surface normal along the x, y, and z coordinate axes, respectively.

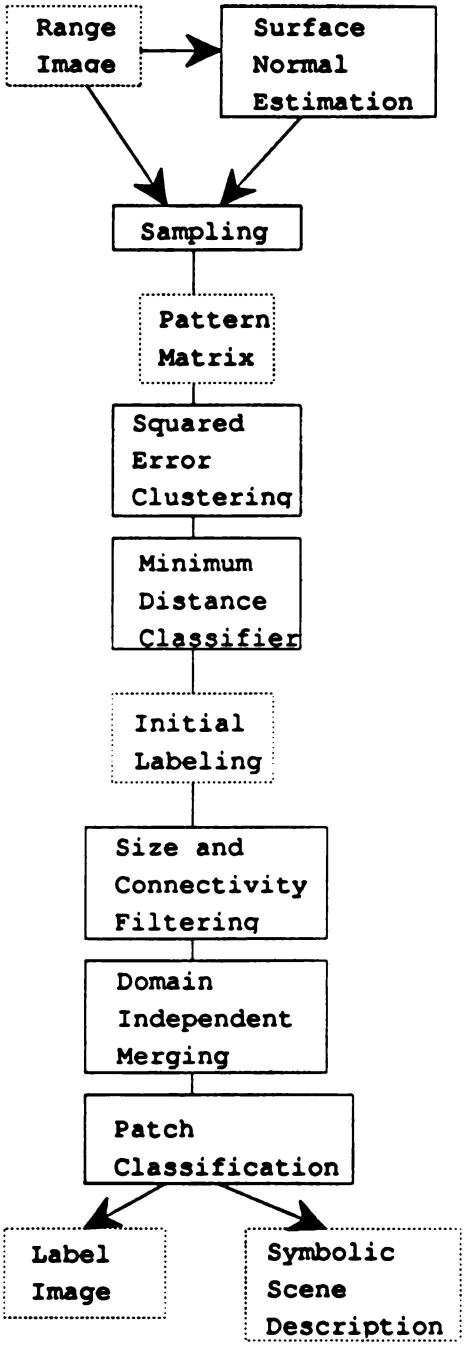
Range Imaqe
Surface Normal Estimation
Sampling
Pattern Matrix
Squared Error Clusterinq
Minimum Distance Classifier
Initial Labeling
Size and Connectivity Filterinq
Domain Independent Merging
Patch Classification
Label Image
Symbolic Scene Description

Figure 4.1: A functional block diagram of the segmentation and classification procedure. Dashed boxes indicate data structures; solid boxes indicate processing steps.

- Our clustering-based segmentation technique uses both the pixel and the corresponding surface normal to assign a segment label. It is convenient to define a *composite image*

  $$\mathcal{C} = \{\mathbf{c}_{ij} = [\mathbf{r}_{ij}\ \mathbf{n}_{ij}]\}$$

  with six measurements at each pixel.

- One of the outputs of the segmentation and classification procedures is a *labeling* of the input image $\mathcal{R}$:

  $$\mathcal{L} = \{l_{ij}\}.$$

  The labels are nominal-valued and take integer values for the sake of convenience. By convention, l_{ij} is zero if the flag value f_{ij} is FALSE. *Intermediate* labelings produced during processing will be denoted $\mathcal{L}_1$, $\mathcal{L}_2$, *etc.*

The goal of segmentation is to obtain a labeling in which regions correspond with the connected 'faces' of the objects in the scene. In our work, objects are composed of smooth surfaces joined at *crease edges* (where the directional derivatives are discontinuous). The faces, then, are the individual patches; we wish the labels to take on the same value for all the pixels belonging to the patch, and different values for pixels outside it. Figure 4.2 shows range and pseudo-intensity images of a scene containing two of our database objects. Later figures will show the intermediate and final results produced from the range image.

4.2.1 Surface Normal Estimation

Local surface orientation has been a popular image feature in range image analysis, and some attention has been paid to the problem of reliably estimating the normal image $\mathcal{N}$ from the range image $\mathcal{R}$. The input depth map $\mathcal{R}$ provides positional data on the scene surfaces, and $\mathcal{N}$ provides estimates of the surface orientation at those points. Orientation is defined locally, but is not intrinsic, since rotation of the object with respect to a fixed coordinate frame rotates the normal vectors as well. Most techniques for estimating surface normals employ the range data in a small neighborhood of the pixel of interest. Furthermore, these techniques make independent estimates at each image pixel for which the normal is desired.

We broadly classify normal estimators as *analytic* or *discrete* [41]:

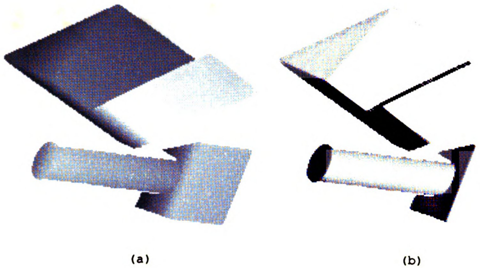

Figure 4.2: Range (a) and pseudo-intensity (b) images of a scene containing two objects.

- *Analytic* approaches fit a plane to the image data in a neighborhood of the pixel of interest and extract the surface normal from the parameters of the fitted plane.
- *Discrete* techniques use subsets of the neighboring image points to obtain several estimates of the surface normal and summarize them, producing a single estimate.

Regression-based analytic techniques have been the most popular in the literature, and our method for surface normal estimation fits in this category. As an example of a competing discrete technique, consider the range point $\mathbf{r}_{ij}$ and the eight pixels surrounding it, forming a 3×3 block. Twenty-eight estimates of the normal at $\mathbf{r}_{ij}$ can be calculated by choosing all $\begin{pmatrix} 8 \\ 2 \end{pmatrix}$ pairs of points from the eight neighbors of $\mathbf{r}_{ij}$ and finding the normal to the triangle formed by the pair and $\mathbf{r}_{ij}$. Presumably, these estimates would be averaged to produce a final value. Heuristics to reduce the computation time and improve the numerical stability of the result would normally be applied. However, a desire for least-squares optimality of the normal vector's defining plane leads one directly to regression techniques.

The regression estimator used to obtain the surface normal should reflect the type and relative power of noise source (or sources) which contaminate the range observations. The approach taken by many researchers is to assume that the measurement error in the range measurement only affects the z coordinate, and that the sampling grid coordinates x and y are error-free. Under this assumption, a linear model for depth at the point of interest $\mathbf{r}_{ij} = [x_{ij}\, y_{ij}\, z_{ij}]$ is

$$z = ax + by + c,$$

where a, b, and c are the parameters to be estimated. Let Ω denote a neighborhood of the range point $\mathbf{r}_{ij}$ containing $k > 2$ 'valid' range measurements. Provided that the k points are not all collinear, the least-squares solution $(\hat{a}, \hat{b}, \hat{c})$ to the system of k equations in 3 unknowns provides an estimate for the normal vector $\mathbf{n}_{ij}$, namely

$$\hat{\mathbf{n}}_{ij} = \frac{(-\hat{a}, -\hat{b}, 1)}{\sqrt{1 + \hat{a}^2 + \hat{b}^2}}.$$

This technique fits a plane to the local surface measurements which minimizes the distance *in the z-value only* from the fitted plane to those measurements.

Not all range sensors provide data which is guaranteed to be error-free in the x and y-coordinates, however. Our sensor, for example, can contain systematic errors in the y coordinate due to misalignment of the laser stripe. The x coordinate is obtained from the position of the motorized stage, and positional errors can be caused by the inherent nonrepeatability of the stepper motor and by quantization effects.

In cases where errors in all three coordinates are to be considered, another least-squares procedure for fitting planes to the local range measurements can be defined using the techniques of principal component analysis [66, 70]. This method proceeds as follows. The 3×3 sample covariance matrix $\mathbf{R} = (r_{kl})$ of the 3D points is obtained. This symmetric matrix has as the diagonal elements r_{11}, r_{22} and r_{33}, the sample variance of the x, y, and z coordinates, respectively. The off-diagonal elements are the sample covariances between pairs of coordinates.

Using standard techniques [102], the three eigenpairs $\{(\lambda_i, \mathbf{v}_i) \mid i = 1, \ldots, 3\}$ of $\mathbf{R}$ can be obtained (λ_i is an eigenvalue of $\mathbf{R}$, and $\mathbf{v}_i$ is the corresponding eigenvector). Without loss of generality, we will assume that the eigenpairs have been sorted so that $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$. Since $\mathbf{R}$ is always nonnegative definite, all eigenvalues are nonnegative.

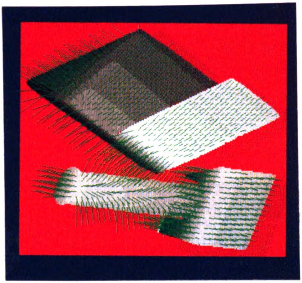

Figure 4.3: Surface normals estimated from the range image of Figure 4.2.

The physical interpretation of the eigenvalues and eigenvectors leads to our estimate of the surface normal. When we normalize our set of (x, y, z) points by subtracting its sample mean, and form a new measurement by projecting the normalized data onto $\mathbf{v}_1$, the sample variance of these projected points is λ_1. Similarly, λ_2 and λ_3 are the sample variances obtained when the data is projected onto the eigenvectors $\mathbf{v}_2$ and $\mathbf{v}_3$, respectively. When we know that our data points are obtained from a locally flat surface, $\mathbf{v}_1$ and $\mathbf{v}_2$ define a plane that approximates the underlying surface. Hence, the eigenvector associated with the *smallest* eigenvalue is an estimate of the surface normal, and λ_3 is the variance of the data with respect to that plane. Additionally, the principal components rotation is optimal in the sense that λ_3 is the *minimum obtainable variance* for a 1-D projection of the data. Geometrically, the plane obtained from $\mathbf{v}_1$ and $\mathbf{v}_2$ is the *minimum perpendicular error* planar fit to the given 3D points.

The above procedure is applied at each pixel in the range image; the output of this process is the normal image $\mathcal{N}$. Figure 4.3 shows the surface normals estimated for the image of Figure 4.2.

4.2.2 Sampling and Clustering

To obtain an initial labeling

$$\mathcal{L}_1 = \{l_{1ij} \mid 1 \leq i \leq m, 1 \leq j \leq n\},$$

we treat segmentation as an unsupervised statistical classification problem (a *clustering* problem) in a six-dimensional feature space. *Cluster analysis* is "the process of classifying [features] into subsets that have meaning in the context of a particular problem" [66]. A full exploration of clustering techniques is beyond the scope of this dissertation; the interested reader is referred to Jain and Dubes [66] for a presentation of techniques with emphasis on computational issues.

Our application of clustering attempts to identify connected image regions corresponding to smooth object surfaces from local geometric features (positions and local orientations). The input to a clustering procedure is an ordered set of feature vectors (typically called *patterns*), and the output of the procedure is one or more *clusterings*, each clustering being an ordered set of labels for the input patterns. The group of patterns having the same label is called a *cluster*. The patterns in our application are six-dimensional elements from the composite image C. This approach to segmentation [64, 57], is often called *local feature clustering*, since the image features involved in clustering are local properties. The (x_{ij}, y_{ij}, z_{ij}) coordinates of image points are useful in clustering because they will help insure that the output clustering consists of *connected* surface patches. The surface normal components $(nx_{ij}, ny_{ij}, nz_{ij})$ attempt to place range pixels with different local orientations into separate clusters. Clearly, these six coordinates constitute a 'minimal' set of features upon which we can obtain a segmentation in which surface patches are connected and segment labels change across significant orientation changes in the image of the object's surface.

In principle, we could input the six-dimensional feature vector at *every* pixel in the composite image C into our clustering procedure, and obtain a label for every pixel directly. Unfortunately, clustering algorithms are expensive in terms of memory consumption and processing time; for this reason, we only cluster a subset of the pixels. This subset is obtained by taking pixels from every kth row and column, with k chosen as small as possible so that the total number of pixels selected from the image does not exceed 1000.

We now discuss the clustering algorithm (called CLUSTER) used for range image segmentation. CLUSTER is a two-phase clustering program which, given an *a priori* upper bound k_{max} on the number of clusters, generates a sequence of clusterings containing 1, 2, ... k_{max} clusters. Like many clustering programs, CLUSTER assigns labels to the input patterns in an attempt to minimize the squared error for the clustering being constructed. For ease of notation, we denote the n six-dimensional feature vectors obtained from the composite image C as $\{\mathbf{c}_1, \ldots, \mathbf{c}_n\}$, with $\mathbf{c}_i = (c_{i1}, \ldots, c_{i6})$ (it is useful to visualize this set of patterns as a $n \times 6$ matrix). Now let the labeling obtained from a k-cluster solution be denoted $\{l_1, \ldots, l_n\}$ (a label for each pixel), and let the k cluster centers be $\mathbf{m}_1, \ldots \mathbf{m}_k$[1], with $\mathbf{m}_k = (m_{k1}, \ldots, m_{k6})$. The squared error is given by

$$E^2 = \sum_{i=1}^{n} \sum_{j=1}^{6} (c_{ij} - m_{l_i j})^2,$$

where l_i is the class label obtained for the ith input pattern $\mathbf{c}_i$.

The major steps in the CLUSTER program are as follows:

1. An initial set of k_{max} clusterings is generated, with the ith clustering containing i clusters. The first clustering places all patterns in one cluster. To obtain the second clustering, two cluster centers are defined. One is the centroid of the patterns, and the other is coincident with the pattern farthest from the first cluster center. The patterns are then labeled with either 1 or 2 depending on which of the two cluster centers it is closest to. A clustering with $j+1$ clusters is obtained from a clustering with j clusters by choosing the pattern farthest from the current clustering as the new cluster center. The squared error is calculated for each clustering in this sequence.

2. A new set of clusterings is created, starting with the k_{max}-cluster solution obtained at the end of Step 1. If any two clusters in that solution can be merged to produce a $(k_{max} - 1)$-cluster solution with lower squared error than the previous $(k_{max} - 1)$-cluster solution, the resultant clustering replaces the previous $(k_{max} - 1)$-cluster solution. Then the $(k_{max} - 1)$-cluster clustering is examined in the same way to produce a (possibly new) $(k_{max} - 2)$-cluster clustering, and so forth.

Steps 1 and 2 are repeated in sequence until none of the k_{max} clusterings change during a pass. The resultant sequence of clusterings is then used as the output of the clustering program, along with the cluster centers and other statistics.

Once a set of clusterings (containing k_{max} solutions with $1, 2, \ldots k_{max}$ clusters) has been obtained from CLUSTER, how is the 'best' clustering chosen? Like Hoffman and Jain, we examine a statistic $M(k)$ developed by Coggins and Jain [32] which reflects the isolation and compactness of the clusters. We specify $k_{max} = 16$ to the

[1] A cluster center is the centroid of the patterns assigned to that cluster.

CLUSTER program and choose the k-cluster solution, where k is as close to k_{max} as possible and $M(k)$ is a local maximum. Choosing a clustering with 'too many' clusters is preferable to choosing a clustering with 'too few' clusters. Subsequent classification steps will not break image regions corresponding to a single cluster; they only merge adjacent regions with similar parameters. Therefore, we are able to accept clusterings which break connected surfaces into more than one cluster.

At this point, we have chosen one of the k_{max} clusterings and therefore, we have a label for each of the input patterns, which are a subset of the image pixels. In addition, we have the k cluster centers $\mathbf{m}_i, i = 1, \ldots, k$. To construct the preliminary labeling $\mathcal{L}_1$, we use a minimum-distance classifier. The label l_{1ij} for composite image pixel $\mathbf{c}_{ij}$ is the index l where the Euclidean distance

$$D = \|\mathbf{m}_l - \mathbf{c}_{ij}\|$$

is smallest; in other words, we assign each pixel the index of its closest (in six dimensions) cluster center.

4.3 Refined Segmentation: Cleaning and Merging

The initial labeling $\mathcal{L}_1$ is often contaminated by the following undesirable artifacts:

- Non-connected patches with the same label. Clustering attempts (through use of the (x, y, z) coordinates of the image points) to create patches which are compact, but connectivity in the image grid is not enforced.
- Patches with an extremely small number of pixels relative to the other image segments.
- Differently-labeled patches belonging to the same object surface. This artifact is known as *oversegmentation*, and is an unavoidable result of our choice of a large value for the number of clusters.

A recursive connected components labeling algorithm is employed to remedy the first problem. It takes the label image $\mathcal{L}_1$ as its input and produces a new labeling $\mathcal{L}_2$ as output; 8-connected regions in $\mathcal{L}_1$ are preserved, but the label values are made

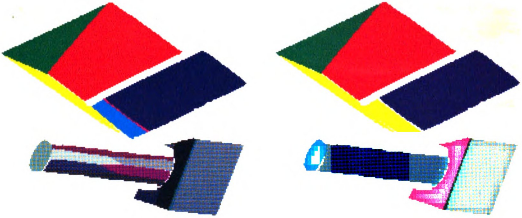

Figure 4.4: Segment labels from clustering and after merging.

unique. After connected component labeling, the number of pixels in each segment is counted, and segments smaller than a threshold value are discarded. Classifications of such segments are frequently unreliable and offer little discriminatory power during the object recognition stage. The output of this cleaning step is a label image $\mathcal{L}_3$. The boundaries between patches are then examined to estimate the change in surface orientation across the boundary, and if the change (averaged over all boundary pixels) is below a threshold value, the patches are merged. Finally, the entire image is relabeled so that the segment numbers begin with 1 and end with k, the number of distinct segments. The output of the relabeling step is $\mathcal{L}_4$.

Figure 4.4 shows the initial segmentation $\mathcal{L}_1$ of the range image in Figure 4.2, and the output $\mathcal{L}_4$ of the merging step. Pixels with the same gray level in the label images belong to the same region.

4.4 Surface Classification

Given a range image and a labeling, our next task is to characterize each segment as planar, spherical, cylindrical, or unknown, and estimate the parameters for the specific surface type. Our recognition system relies on the geometric parameters of image segments to generate and validate hypotheses of identity and position for the objects in the image. We place the geometric parameters of our surface types into three classes: orientation parameters, location parameters, and intrinsic parameters.

- *Orientation parameters* convey information about the orientation of the surface under consideration. For example, the plane's orientation parameter is its normal vector. Spheres do not possess an orientation parameter. A cylinder's orientation parameter is a vector in the direction of its axis.

- *Location parameters* convey information about the position of the surface in 3D. For example, the plane's location parameter is a scalar, namely the minimum (perpendicular) distance to the origin from the plane. A sphere's location parameter is the 3D point at its center. For a cylinder, we take the location parameter to be *any* point on its axis.

- *Intrinsic parameters* are surface features that do not depend on the position or orientation of the object. Planes have no miscellaneous parameters, since they are completely specified by their normal (an orientation parameter) and their distance from the origin (a location parameter). Spheres and cylinders have one miscellaneous parameter, the radius.

Why should we estimate these particular parameters, and why are they important? The domain of surfaces which can be completely characterized by these three families of parameters includes all of the surfaces of revolution, including the three surface types we employ. So, these parameters are both intuitive and descriptive. This approach can be extended to other families of surfaces (for example, generalized cylinders with straight axes). It should be noted that more complicated surfaces will have more intrinsic parameters (and possibly more orientation parameters), providing additional constraints to be used at recognition time. In Chapter 5, we will construct predicates from orientation and intrinsic parameters; they will allow us to discard incorrect hypotheses of object identity. The location and orientation parameters can also be used to constrain the degrees of freedom in an estimate of the location and orientation of the object as a whole. These parameters are relatively easy to extract from the segmented range image using appropriate methods.

The hierarchical classification procedure is shown in Figure 4.5. A plane is fit to the 3D points in the segment, and accepted if the mean-squared-error of the fit is

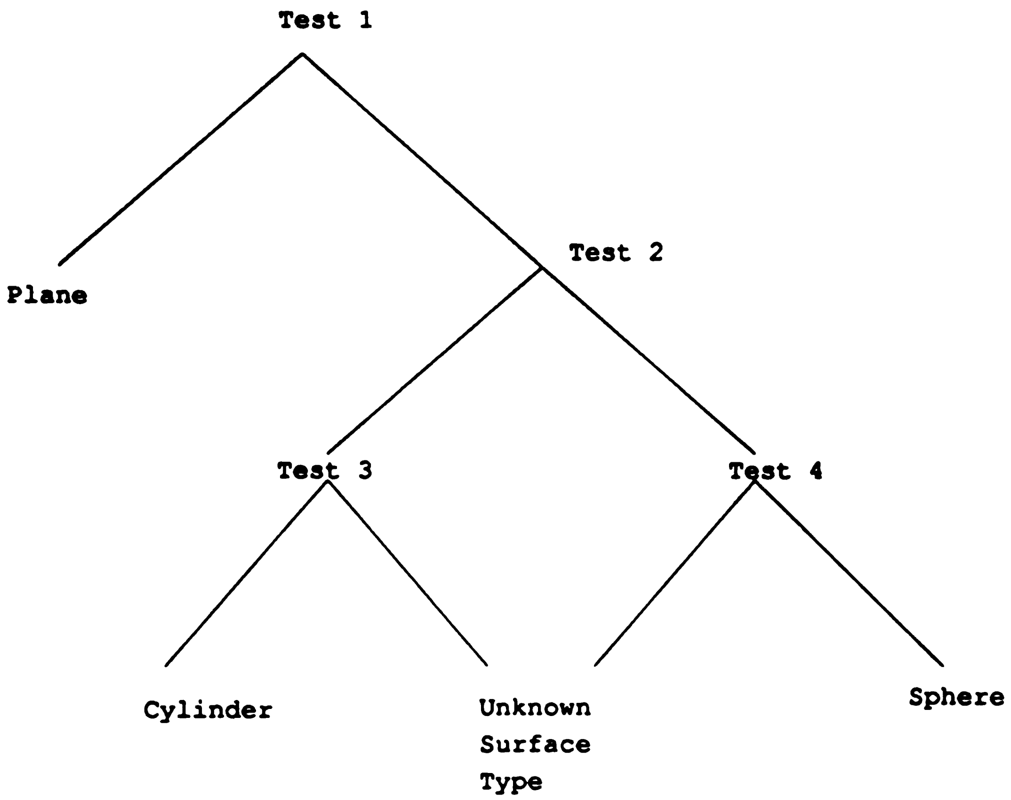

Figure 4.5: Decision tree for surface classification.

below an empirically-determined threshold. If the threshold is exceeded, however, we use the minimum curvature direction and corresponding maximum curvature value to estimate the orientation, location, and radius parameters for a hypothesized cylindrical patch. A squared-error statistic is also calculated for this hypothesis; if it exceeds a threshold, the cylindrical fit is rejected, and the patch is labeled as 'unknown'. If, after the planar hypothesis fails, the minimum curvature is *not* close to zero, we attempt a spherical fit using a nonlinear optimization algorithm, and again calculate a squared-error statistic. If the statistic lies below a threshold, the spherical fit is accepted; otherwise, the patch is labeled as 'unknown.' The classified image surfaces are expressed as LISP S-expressions for use by the recognition module (described in Chapter 5).

4.4.1 Testing for Planes

The strategy used to estimate surface normals is again used here; the primary difference between these two applications of the principal components fit is the number of points used in the fit. Earlier, we used points in a small neighborhood (5 × 5) around a range pixel of interest. Here, we use *all* of the range pixels in the image segment (typically several hundred 3D points). Given the set of 3D points, we find the sample covariance matrix **R** as before, and extract its eigenvectors and corresponding eigenvalues. The eigenvector $\mathbf{v}_3 = (v_{31}, v_{32}, v_{33})$ corresponding to the smallest eigenvalue λ_3 is normal to the best-fitting plane (in a squared-perpendicular-distance sense). The coefficients v_{3i} provide the orientation parameter for the plane. We then estimate the location parameter by finding the estimated minimum distance between the plane and the origin. The implicit equation of the fitted plane is given by

$$v_{31}x + v_{32}y + v_{33}z + d = 0, \tag{4.1}$$

where d is the unknown location parameter to be determined. The estimate $\hat{d}$ is obtained by plugging the centroid $(\bar{x}, \bar{y}, \bar{z})$ of the given 3D points into Equation (4.1), producing

$$\hat{d} = -(v_{31}\bar{x} + v_{32}\bar{y} + v_{33}\bar{z}).$$

The validity of the planar fit is tested by comparing the RMS (root mean-square) error of fit E_{rms} for the plane to a predetermined threshold T_{plane}. E_{rms} is given by

$$E_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_{31}x_i + v_{32}y_i + v_{33}z_i + \hat{d})^2}.$$

We do not have to calculate E_{rms} explicitly; it is simply $\sqrt{\lambda_3}$. Note that the units of E_{rms} are length (in our case, inches). We chose a threshold value of 0.05 inches for the planar fit. Our first surface classification test, then, is

> **Test 1**: IF $E_{rms} \leq T_{plane}$, classify the surface as **planar** with the estimated orientation and location parameters.

In our experiments, T_{plane} was chosen empirically. However, if the noise contaminating the range data follows a multivariate normal distribution with a diagonal covariance matrix, a χ^2 test for planarity can be developed [39]. However, the multivariate normal assumption does not hold for our range data, and since the threshold is dimensioned as length, we obtain empirical thresholds from *a priori* assumptions about the type and parameters of the surfaces to be encountered. The threshold value used in our experiments was $T_{plane} = 0.05$ inch. This test would classify very gently curved surfaces (say, spheres with a radius of 20 inches) as planar; but if we employ realistic assumptions about the surfaces to be expected, we can 'tune' this threshold (and the others used in the segmentation procedure) for the expected set of objects.

4.4.2 Estimating Curvature

The remaining tests for surface classification use surface curvatures and their associated directions to estimate the parameters of the various surface types. This section discusses our approach to curvature estimation; a more detailed discussion of several methods appears in [41]. Curvature estimation has been used extensively in range image analysis; most researchers have used curvature signs to make *qualitative* decisions about surface type [10, 36, 122, 64]. Only recently have researchers attempted to infer *quantitative* properties (specifically, the geometric parameters of quadrics) from curvature [14]. Quantitative interpretations allow for more detailed classifications of surfaces; however, the noisiness of estimated curvatures makes contamination of parameter estimates more likely.

Surface curvatures are intrinsic properties defined at a point on a C^2-continuous surface. Focusing on an interior point $\mathbf{p}$, there exists a tangent plane $T_{\mathbf{p}}$ and an infinite number of orthonormal basis vector pairs for $T_{\mathbf{p}}$. Figure 4.6 shows a surface, the global coordinate system, the tangent plane at a point, and one choice of an orthogonal basis for the tangent plane. How does the surface curve at $\mathbf{p}$ when we travel in a certain direction in $T_{\mathbf{p}}$? The direction of travel defines a curve embedded in the surface. Except in cases where the direction of travel does not affect the surface curvature (*e.g.* a plane, where curvature is everywhere zero, or a sphere, where the curvature is everywhere the inverse of the radius), there is an unique pair of directions, *orthogonal in the tangent plane*, for which the surface curvature attains a maximum in one direction and a minimum in the other. These directions are called *principal directions*, and their associated surface curvatures are the *principal curvatures*; they will be denoted $\mathbf{v}_{min}, \mathbf{v}_{max}$, and k_{min}, k_{max}, respectively.

The surface types used in this work have some interesting curvature properties. As mentioned above, the principal curvatures of planar surfaces are everywhere zero,

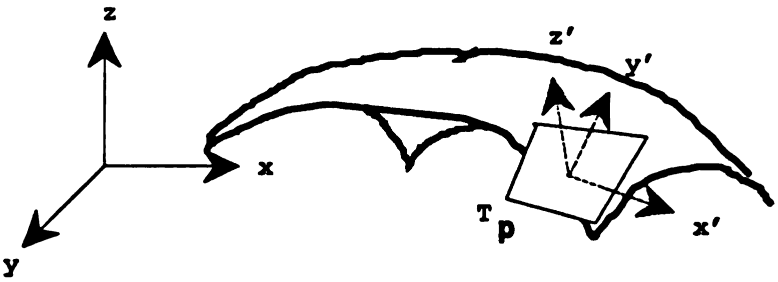

Figure 4.6: Sensor and local coordinate systems.

and the direction vectors are not uniquely defined. On spherical surfaces, the curvatures are equal to the inverse of the sphere's radius, and again, the directions are not uniquely defined. On a cylindrical surface, however, the directions are uniquely defined, and useful for parameter estimation. The absolute maximum curvature is equal to the inverse of the cylinder's radius, and the corresponding direction is perpendicular to the cylinder's axis. The minimum curvature is zero, and the corresponding direction is parallel to the axis. As a matter of fact, all surfaces of revolution have a similar property; one principal direction is always perpendicular to the axis (a *parallel* of the surface), and the other direction is always orthogonal to the axis (a *meridian* of the surface).

Our strategy for curvature estimation is an *analytic* technique [41]: at each pixel of interest $\mathbf{p}$, curvature is estimated by

- extracting the neighborhood $\Omega_\mathbf{p}$ of p,
- centering the data in the neighborhood to place $\mathbf{p}$ at the origin,
- estimating the tangent plane $T_\mathbf{p}$,
- fitting a bicubic surface to $\mathbf{p}$ and its neighbors (using the local coordinate system defined by $T_\mathbf{p}$ and the surface normal), and
- extracting the principal curvatures and directions from the coefficients of the fitted surface.

We now explain the details of each step.

Extraction of the neighborhood $\Omega_\mathbf{p}$ of the range pixel $\mathbf{p}$ is easily performed since the image is in 'raster-scan' format; we can simply create a list of the 3D coordinates of the pixels in a $k \times k$ neighborhood (k odd) centered at $\mathbf{p}$. The size k governs the accuracy of the curvature estimate. First of all, we require $k > 3$ to ensure that the system of equations solved to fit the bicubic surface is overdetermined. In our work, we choose $k = 9$. Small values of k involve fewer pixels in the estimates, and variation due to measurement errors is more pronounced. Larger neighborhoods perform more implicit smoothing of the range data as part of the regression process, and variation of the curvature values (and their associated directions) due to noise will be reduced. A large neighborhood size will also reduce the number of estimates produced from the segment, because *we insist that* $\Omega_{\mathbf{p}}$ *be a* full *neighborhood*, *i.e.* if any of the range pixels in $\Omega_{\mathbf{p}}$ are invalid due to shadowing or edge effects, then a curvature estimate will not be produced at that pixel. Through this restriction, we are ensuring that estimates are produced only in the interior of the segment. Since we are using these estimates to find parameters of curved image surfaces, the interior values, in our opinion, are the most reliable; curvatures estimated near physical or shadow edges should not be allowed to contaminate the ensemble of estimates within the image surfaces.

A centering step is applied to the list of 3D points in order to place the pixel of interest $\mathbf{p}$ at the origin. This centering allows partial derivatives of the local surface fit (see below) to be computed more efficiently. We then obtain a *local coordinate system* (x', y', z') for the data using the principal components projection described earlier to estimate surface normals. This procedure essentially estimates the tangent plane $T_{\mathbf{p}}$ and projects the data into a new coordinate system defined by the three eigenvectors extracted as part of the principal components process. The subsequent regression steps perform their fits with respect to this new coordinate system. This projection is illustrated in Figure 4.6, which shows a curved surface with range measurements made in a sensor coordinate frame (x, y, z), the tangent plane $T_{\mathbf{p}}$ at a range pixel of interest $\mathbf{p}$, and the basis vectors for the local coordinate frame (x', y', z'). We choose the local coordinate system so that the x' and y' direction vectors (denoted $\mathbf{v}_{x'}$ and $\mathbf{v}_{y'}$) span the tangent plane, and z' is the direction normal to $T_{\mathbf{p}}$. Note that the center pixel remains at (0,0,0) in this new coordinate system, since the transformation is a rigid rotation.

At this point we have a list of k^2 3D points $\{(x'_i, y'_i, z'_i)\}$ in the local coordinate frame defined at $\mathbf{p} = (0, 0, 0)$. We then solve the linear least squares problem associated with the bicubic surface fit

$$z' = f(x',y') = a_1x'^3+a_2x'^2y'+a_3x'y'^2+a_4y'^3+a_5x'^2+a_6x'y'+a_7y'^2+a_8x'+a_9y'+a_{10}.$$

When we plug each of the k^2 3D points into this equation, we obtain a system of k^2 equations in the 10 unknowns a_i. The least-squares solution can be obtained using standard techniques; we employ routines in the LINPACK linear algebra software library [33].

The surface fit to the neighborhood of $\mathbf{p}$ is commonly called a *graph surface*, since it is the graph of the function $f(x',y')$. The mean and Gaussian curvatures are related to the two invariants of the *Hessian matrix* of the graph surface function f [100]:

$$\mathbf{H} = \begin{bmatrix} f_{x'x'} & f_{x'y'} \\ f_{y'x'} & f_{y'y'} \end{bmatrix} = \begin{bmatrix} 2a & 2b \\ 2b & 2c \end{bmatrix}$$

Gaussian curvature K is the determinant of $\mathbf{H}$, and the mean curvature H is one-half the trace of $\mathbf{H}$. The two principal curvatures are obtained from H and K by

$$\begin{aligned} k_1 &= H + \sqrt{H^2 - K} \\ k_2 &= H - \sqrt{H^2 - K}. \end{aligned} \tag{4.2}$$

The required partial derivatives are easily obtained from the graph function f; their values at $\mathbf{p} = (0,0,0)$ are given by

$$\begin{aligned} f_{x'x'}(0,0) &= 2a_5 \\ f_{y'y'}(0,0) &= 2a_7 \\ f_{x'y'}(0,0) &= a_6 \\ f_{x'}(0,0) &= a_8 \\ f_{y'}(0,0) &= a_9 \end{aligned} \tag{4.3}$$

In addition to the curvatures themselves, we are interested in their direction vectors. These vectors, expressed as a linear combination of the x' and y' basis vectors $\mathbf{v}_{x'}$ and $\mathbf{v}_{y'}$, are given by

$$\begin{aligned} \mathbf{v}_1 &= \begin{cases} (a-c+\sqrt{(a-c)^2+4b^2})\mathbf{v}_{x'} + 2b\mathbf{v}_{y'}, & \text{if } a \geq c, \\ 2b\mathbf{v}_{x'} + -(a-c-\sqrt{(a-c)^2+4b^2})\mathbf{v}_{y'}, & \text{otherwise} \end{cases} \\ \mathbf{v}_1 &= \begin{cases} -2b\mathbf{v}_{x'} + (a-c-\sqrt{(a-c)^2+4b^2})\mathbf{v}_{y'}, & \text{if } a \geq c, \\ (a-c-\sqrt{(a-c)^2+4b^2})\mathbf{v}_{x'} + 2b\mathbf{v}_{y'}, & \text{otherwise} \end{cases} \end{aligned} \tag{4.4}$$

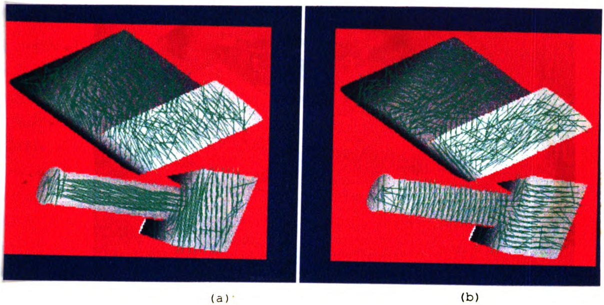

Figure 4.7: Minimum (a) and maximum (b) curvature directions for the range image of Figure 4.2.

At this point, we must clarify our notion of minimum and maximum curvatures. The sign of a principal curvature indicates the 'direction of bending' of a surface, but does not influence its geometric parameters. For example, the principal curvature of a cylinder in a direction perpendicular to the axis (and in the tangent plane) can be either the inverse of the cylinder's radius or the negative of the inverse of the radius, depending on whether we are estimating curvature on the 'inside' or the 'outside,' respectively. If we decide that the surface is indeed cylindrical, all we need to examine are the *absolute* values of the principal curvatures: the signs are discarded. For that reason, we will define the *maximum principal curvature* k_{max} as the maximum ***absolute*** principal curvature. Similarly, k_{min}, the ***minimum principal curvature***, will refer to the minimum absolute principal curvature. This convention allows us to test for cylinders *versus* spheres by examining the minimum curvature, without regard to the effects of the curvature signs. For consistency, we will denote the principal directions corresponding to k_{min} and k_{max} as $\mathbf{v}_{min}$ and $\mathbf{v}_{max}$, respectively.

Figure 4.7 shows minimum and maximum curvature directions for the image of Figure 4.2.

4.4.3 Testing and Parameter Estimation for Cylinders

Recall that Test 1 was used to decide whether a segment in the range image was planar or nonplanar. If the results of this test suggest that the surface is nonplanar, we next examine the ensemble of curvature estimates and directions to classify the segment as cylindrical, spherical, or 'unknown'. How do cylinders differ from spheres in terms of curvature? If one travels on a cylinder in a direction parallel to the axis, the surface does not curve. Formally, the minimum (absolute) principal curvature is zero, and the corresponding direction is parallel to the axis. Given the normal and the minimum curvature direction, the maximum curvature direction is uniquely determined; the corresponding principal curvature is the inverse of the radius. In contrast, all points on a spherical surface are *umbilics*, where the principal curvatures are equal and the principal directions are not uniquely defined. Hence, a simple test for cylindrical *versus* spherical patches is to compare a central estimate of the minimum curvature against zero. When we are dealing with noisy data, the curvature estimates will never be exactly zero; therefore, an empirically-determined threshold is used. In previous work [41], we employed the median of curvature estimates on a smooth patch; this was an attempt to avoid the effects of outlying estimates. Since a reasonably-sized segment in a segmented range image will have several hundred pixels, we will obtain several hundred curvature estimates. In our opinion, the median is preferable to the mean, and may be preferable to trimmed averages. An ensemble of curvature estimates obtained from a single smooth surface is usually contaminated with a large number of outlying estimates. All of the surface types considered in this thesis have constant principal curvatures, so the median has been the most reliable estimator of the principal curvature of the entire patch.

> **Test 2**: IF $k_{min}^{median} < T_{cylinder}^{k}$, attempt to fit a cylinder to the surface; otherwise, attempt to fit a sphere.

In our experiments, $T_{cylinder}$ is 0.15 inches. In theory, this would allow us to classify spheres with radii up to 6.66 inches.

Assume that Test 2 is passed, and therefore we wish to fit a cylinder to the segment. We can obtain estimates of the orientation, location, and radius parameters by examining the minimum curvature direction $\mathbf{v}_{min}$, the surface normal, and the maximum curvature estimate k_{max}, all estimated at the surface point producing the median minimum curvature estimate. Recall that the orientation parameter of a cylinder is a vector in the direction of the axis. This vector should be parallel to the minimum curvature direction estimates; therefore, we use $\mathbf{v}_{min}$ as an estimate of the axis direction. The radius of the cylinder is taken as $\hat{r} = \frac{1}{k_{max}}$.

Estimation of the location parameter (any point on the axis) presents a problem. Given a point $\mathbf{p}$ on the surface, the radius estimate $\hat{r}$, and the surface normal $\mathbf{n_p}$, a point on the axis could have either of the two values

$$\mathbf{p} \pm \hat{r}\mathbf{n_p},$$

depending on whether the surface is convex or concave. Here, we test both cases. The RMS error test described below is applied under each assumption; if the surface is indeed cylindrical, one of the two cases will produce a smaller error. We accept the 'better' solution, conditioned on its satisfying our threshold test. We have never encountered an image in which *both* hypotheses satisfy the RMS criterion.

To test the estimated cylindrical fit, we calculate the average sum of squared distances from the observed data points to the hypothesized cylinder whose parameters were given above. The error for each individual pixel is calculated as follows. The minimum distance between the hypothesized axis (determined by the location and orientation parameters) and the observed point is calculated; it is the length of the line segment joining the point and the line, and perpendicular to the line. The difference between this value and the predicted radius is the error. Our overall error of fit is the square root of the average squared error for each pixel.

> **Test 3**: If the error measure lies below $T^{RMS}_{cylinder}$, classify the patch as a cylinder with the estimated orientation, location, and radius.

Recall that this test is applied twice: once on the hypothesis that the cylinder is 'negative' (the surface is concave as viewed), and once on the hypothesis that the cylinder is positive (convex when viewed). If the RMS error computed for either of these hypotheses is below the threshold, the surface classification is accepted, with the parameters of the correct hypothesis. If neither error measure is below $T^{RMS}_{cylinder}$, however, the surface is labeled as an 'unknown' type.

4.4.4 Testing and Parameter Estimation for Spheres

If the median minimum curvature test (Test 2) indicates that the surface is not 'locally flat' in one direction in the tangent plane, we attempt to fit a sphere to the 3D points in the segment. The technique used was described in [39]. Points (x, y, z) on a sphere of radius r and center location (x_0, y_0, z_0) satisfy

$$f_{sphere}(x, y, z; r, x_0, y_0, z_0) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 - r^2 = 0$$

This implicit form lends itself to solution via either linear or nonlinear least squares methods. We use a nonlinear technique here; it allows us to adjust the geometric parameters directly to optimize the fit to the data, whereas linear methods would adjust parameters which are a mixture of the geometric parameters.

The squared perpendicular distance between a point $\mathbf{p} = (x, y, z)$ and the hypothesized sphere is

$$E_{\mathbf{p}}^2 = f_{sphere}^2(x, y, z; r, x_0, y_0, z_0)$$

The parameter vector which minimizes $\sum_{\mathbf{p}} E_{\mathbf{p}}^2$ is obtained using the Levenberg-Marquardt nonlinear optimization technique [101, 92]. This method requires formulas for the partial derivatives of the objective function $E_{\mathbf{p}}^2$ with respect to the geometric parameters. Like many nonlinear optimization methods, the Levenberg-Marquardt procedure is iterative, and requires an initial estimate of the parameter vector. As an initial estimate of the radius, we provide the inverse of the median minimum principal curvature. By projecting the corresponding point 'into' the sphere along the estimated surface normal, we can also obtain an estimate of the center location. However, the spherical objective function seems to be particularly easy to optimize (and indeed it has only one minimum), so any reasonable estimate would probably work as well; only the speed of convergence would be affected.

Twenty-five iterations through the optimization procedure have proven to be sufficient to accurately fit a sphere. As a matter of fact, once the overall error measure $\sum_{\mathbf{p}} E_{\mathbf{p}}^2$ has either dropped below a threshold value, or changed very little between two iterations, the procedure terminates. After termination, we examine the overall error measure to judge the goodness of fit.

> **Test 4**: If $\sqrt{\sum_{\mathbf{p}} E_{\mathbf{p}}^2} < T_{sphere}$, then classify the segment as a sphere with the given parameters.

If Test 4 is 'failed', we label the segment as 'unknown' and do not attempt a further classification of that segment.

4.5 Examples

The classification procedures were applied to the eight segments (see Figure 4.4) obtained from the range image of Figure 4.2. The LISP code created from the classification procedure appears in Figure A.2 of Section A.2 of the Appendix. Summarizing the output, we obtained

- A plane with implicit equation $0.174x + 0.121y + 0.977z - 1.299 = 0$ and area of 4.75 square inches,
- A plane with implicit equation $-0.612x + 0.426y + 0.666z + 1.536 = 0$ and area of 2.30 square inches,
- A plane with implicit equation $0.170x + 0.122y + 0.978z + 0.291 = 0$ and area of 3.82 square inches,
- A plane with implicit equation $-0.775x - 0.591y + 0.223z - 1.482z = 0$ and area of 4.47 square inches,
- A plane with implicit equation $-0.934x + 0.207y + 0.289z - 1.346 = 0$ and area of 2.63 square inches,
- A plane with implicit equation $-0.930x + 0.209y + 0.301z - 1.346 = 0$ and area of 0.93 square inches,
- A plane with implicit equation $0.303x - 0.068y + 0.950z - 0.384 = 0$ and area of 0.04 square inches, and
- A cylinder with radius 0.660, direction vector $(-0.921, 0.224, 0.319)$, a point $(0.355, 0.534, -1.862)$ on the axis of symmetry, and area of 2.48 square inches.

A Sun 4/280 computer required 5 minutes of CPU time to produce the segmentation and classification output from the range image.

Figure 4.8 shows the range, pseudo-intensity, surface normal, initial segmentation, and final segmentation images for a scene with two objects (and significant occlusion). The LISP S-expressions generated from the classified image appear in Figure A.3 in Section A.2 of the Appendix.

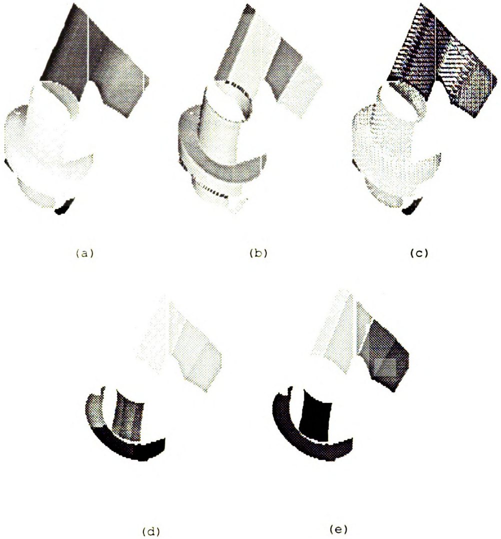

Figure 4.8: Example of segmentation and classification procedures. (a) Range image. (b) Shaded image. (c) Surface normals. (d) Initial segmentation from clustering. (e) Final segmentation.

4.6 Summary

This chapter has described a technique for range image segmentation and classification. Segmentation is performed by a sequential split-and-merge procedure based on squared-error clustering. Classification employs regression, curvature estimation, and nonlinear optimization. The output of the segmentation and classification procedure is a label image and a symbolic scene description (LISP S-expressions).

We consider the segmentation procedure to be the weakest link in the overall object recognition and localization system. One major drawback is its inability to classify more complicated surfaces such as cones, other general surfaces of revolution, or developable surfaces (*i.e.* cylinders with non-elliptical cross sections). Our recognition module can handle many such surfaces as long as they can be characterized by orientation, location, and intrinsic parameters. However, robust detectors for such surfaces are difficult to develop.

The other drawback of this segmentation and classification module is the lack of a feedback path from the model database. Once segments are identified, surfaces are classified using a data-driven approach. In the manufacturing environment, however, we can often state *a priori* that only a certain set of objects will be sensed. If we have geometric models for this allowable set of parts, a *model-driven* segmentation procedure can exploit the geometric knowledge in our manufacturing database to provide constraints on segmentation. The benefits of such feedback extend beyond classification, however. If we identify a scene surface as one of a small number of model surfaces from different objects, the very act of classification has suggested candidate scene interpretations. We have implemented a very simple model database pruning procedure based on this idea in Chapter 5, but the propagation of interpretation constraints from classification to recognition (in the context of model-driven classification) may provide powerful new techniques for scene interpretation.

Chapter 5

3D Object Recognition and Localization: The BONSAI System

bon · sai \ (')bōn-'sī, 'bōn-,, 'bän-, *also* 'bän-,zī \ *n, pl* **bonsai** [Jp] (ca. 1929): a potted plant (as a tree) dwarfed by special methods of culture; *also*: the art of growing such a plant.
—from *Webster's Ninth New Collegiate Dictionary*

5.1 Introduction

This chapter describes BONSAI, the recognition module of the model-based vision system developed in this thesis. Relational graphs extracted from CAD models are compared with the scene graph returned by the segmentation and classification modules, with object identities and estimates of location and orientation (pose) as output.

BONSAI recognizes objects by searching an Interpretation Tree (IT). The search paradigm maps correspondences between a subset of scene features and a subset of model features into a tree structure. Nodes in the IT represent *interpretations*

$$\mathcal{I} =< \texttt{object}, \{(S_1, M_{l_1}), \ldots, (S_k, M_{l_k})\}, (\mathbf{R}, \mathbf{t}) >,$$

containing one or more *associations* (S_i, M_j) between single scene entities and single model entities[1]. Not all associations between model and scene entities are possible; predicates applied to interpretations allow BONSAI to reject inconsistent hypotheses.

Figure 5.1 illustrates the search process for a simple 2D object. The object model consists of several line segments and one circular arc, labeled M_1 through M_8, which (together with their parameters and interrelationships) describe the silhouette of the

[1] `object` identifies the object model under consideration, and **R** and **t** are estimates of the rotation and translation; see Section 5.3.

object. The scene description also consists of line segments and a circular arc, labeled S_1 through S_6[2]. However, some model segments are missing and others have been partially occluded. Gray boxes mark tree nodes where one of four predicates (logical functions of an interpretation) indicate that the interpretation is invalid. When an interpretation fails one or more predicates, search continues with a different association. In the example, only one interpretation involving all the scene entities satisfies the predicates.

In general, however, a successful interpretation need not associate every scene entity with a model entity; indeed, sometimes we must *avoid* interpreting a scene entity, because it lies on a different object than the one currently under consideration (*e.g.* in multi-object scenes). Scenes containing such 'spurious' entities are called *cluttered*; the presence of clutter has a dramatic impact on the computational complexity of IT search. Prior theoretical work [45, 46] has shown that in cluttered environments (where scene entities belonging to the objects of interest are mixed with entities from other objects), the combinatorics of the search process can make a naive implementation of IT search impractical. BONSAI's solution is to use the search procedure only long enough to obtain a *minimal* hypothesis of identity, namely one which produces an unique pose estimate. This estimate is then fed to a cooperative module which attempts to verify the hypothesis by synthesizing an image of the object and a segmentation from the CAD database, and comparing the synthetic image to the input scene. If the input and synthetic images are similar, the synthetic segmentation accompanying the synthetic range image is compared against the scene segmentation, in order to extend the hypothesis (by adding additional associations) without going back to the search procedure. The extended hypothesis is then re-verified. In effect, the recognition module creates very small interpretation trees, only a few levels deep, preferring to spend time verifying candidate matches rather than searching deeper for associations (this observation inspired our choice of the name BONSAI for the system).

The chapter is organized as follows. Section 5.2 surveys related work on 3D object recognition. The interpretation tree paradigm is also introduced (in a general form) there; Section 5.3 specializes the concept of the interpretation tree to our environment,

[2] We assume that the scene entities have been sorted; this ordering defines the sequence of interpretations. The sorting step can greatly influence the performance of the recognition procedure. We address this problem in a later section.

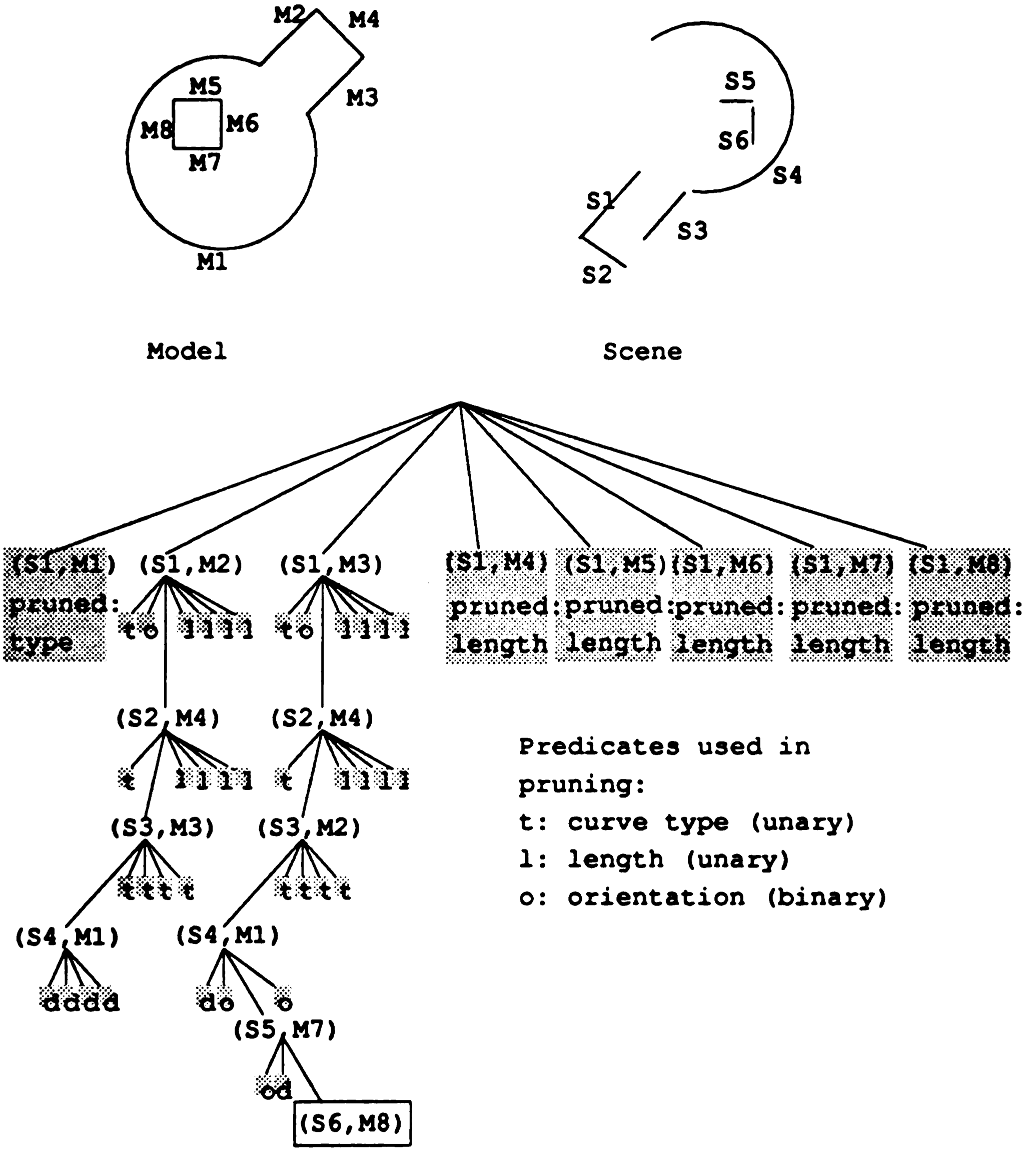

Figure 5.1: Example of IT search for a 2D object.

and gives a high-level description of BONSAI's recognition approach. Section 5.3.7 discusses the system's technique for selecting a subset of models from the database to attempt to match with the scene description. Section 5.3.8 details the steps in interpretation *after* the selection of a particular candidate model; particular attention is paid to the geometric predicates used to prune the search process. Once the interpretation has adequately constrained a pose transformation, the hypothesis is tested using synthetic imagery. Details are discussed in Section 5.5. Section 5.6 discusses interpretation tree search as sequential tests of hypotheses and explores error models. Simulations for estimating the average time complexity of search are also discussed. Remarks on timing, the potential for parallel implementation, and other summary comments appear in Section 5.7. Recognition results produced by this system are described in Chapter 6.

5.2 Survey

This section describes several model-based object recognition systems which have appeared in the computer vision literature. Some of these systems have employed CAD models, and others use alternative representations. Table 5.1 contains a summary of each system in terms of database size (including the number of objects in the published papers), sensing modality, model source, matching strategy, allowable object surfaces, and the presence of occlusion in the scene. Our survey emphasizes complete systems and 3D-3D matching methods, and is by no means exhaustive; the reader should refer to Besl and Jain [9], and Brady *et al.* [18] for more complete surveys. The papers cited below also contain surveys of prior work.

Making comparisons between systems in terms of "recognition power," "speed," or "accuracy" is a difficult task. Often, design assumptions are implied rather than stated explicitly, complexity analyses are avoided, and noise models are ignored or incompletely specified. Unlike many fields, the computer vision community has yet to agree on a database of realistic, representative images to test algorithms. While it is important to avoid straightjacketing the freedom of researchers to pursue unconventional strategies in sçene interpretation, the results they obtain *must* be placed in context in order to be useful. The development of standards for evaluation of image analysis systems will be a significant step toward wider acceptance of such systems outside the research community.

Table 5.1: Object Recognition Systems

Citation	# Objects in Database	Sensing Modality	Model Source	Matching Strategy	Surface Types Used	Occlusion Present
Brooks [21]	several	intensity	CAD	constraint propagation	generalized cylinder	no
Ikeuchi [62]	1	photometric stereo	CAD	search	cylinder, plane	yes
Bolles [15]	1	range	CAD	search	cylinder, plane	yes
Hansen [51]	1	range	CAD	search	plane	no
Kak [71]	2	range	CAD	graph matching	plane, cylinder, sphere, torus	yes
Chen [27]	2	range	multiple views	search	plane,cylinder sphere	yes
Grimson [47]	1	range	hand built	search	plane	yes
Hoffman [55]	10	range	CAD	evidence based	3 qualitative types	no
Vemuri [118]	6	range	multiple views	alignment	5 qualitative types	no
Fan [37]	10	range	multiple views	graph matching	quadrics	yes
Chen [28]	3	range	hand built	search	4 qualitative types	yes
Lowe [78]	1	intensity	hand built	optimization	points/lines	yes
Huttenlocher [61]	1	intensity	hand built	alignment	point-based	yes

CAD-based vision systems vary widely. Some researchers (*e.g.* Brooks [21]) implemented their own geometric modeler and developed the model-based vision system on that platform. Other work [51, 79, 71] uses experimental modelers which are not commercial products. Systems based on experimental modelers can be specialized to take advantage of the unique features of that modeler, but such specialization limits the ease of technology transfer, since manufacturing enterprises *do not* (in general) *use experimental modelers.* In order for CAD-based vision systems to be accepted in the marketplace, they must adopt the prevailing standards for geometric representation. To our knowledge, our system is the only one which does so.

One of the earliest model-based 3D object recognition systems was ACRONYM [21, 20]. This system laid the foundation for much of the subsequent work in 3D object recognition. In ACRONYM, objects are represented as a hierarchy of generalized cylinders, and models are constructed using a custom-built geometric modeler. In addition to models for specific objects, *classes* of models can be built from the individuals. Relationships between class members are embedded in a *restriction graph*, and the *object graph* contains individual model primitives and relations between them.

Examples of classes and class members described in the papers are wide-bodied aircraft, with members 747 (further subclassified as 747B and 747SP) and L-1011, .and electric motors, with members gas pump, industrial motor, and carbonator motor. Within-class variations are embodied in constraints on the parameters applicable to that class. Variations are of three types: size, structure (presence or absence of a subpart), and relationships (relative positioning of subparts).

Generalized cylinders project as strips ('ribbons') in the 2D image plane, and their ends (assumed in ACRONYM to be circular) project as ellipses. A line finder is used to extract ribbons and ellipses from the image, and these entities serve as input to the image interpretation system. Predictions of image entities which are invariant over a large range of camera and model parameters are made and matched against the extracted image artifacts. These local matches are combined in a fashion which satisfies the existing model constraints, thus producing as refined an identification as possible, along with pose information. Besl and Jain [9] observe that this system can be misled if the low-level input (linear structures) is of consistently poor quality. Also, ACRONYM as described uses only edge information. It is possible that performance could be improved if predictions could be made about the allowable surface types in the input image. This might make it easier to incorporate range information into the sensed data used by ACRONYM.

Ikeuchi [62] has proposed an off-line method for generation of an interpretation tree from CAD models of objects composed of planar and cylindrical faces. Objects are represented as a collection of prototypical views (equivalence classes), and small changes in pose within a view class are described as *linear shape changes*, in which the topology of a line drawing describing the object does not change. A *nonlinear* shape change occurs between two line drawings from different equivalence classes.

Ikeuchi's interpretation tree represents possible associations of model entities with scene entities. A fully-instantiated IT enumerates all possible assignments, and is computationally burdensome to construct. In Ikeuchi's paper, the interpretation tree is constructed as follows. A set of sixty view directions corresponding to the patches on a tessellated sphere surrounding the object are defined. From each view direction a synthetic image of the object is generated. Those surface patches (cylindrical and planar) which satisfy some lighting and size constraints are labeled as *observable* for that view[3]. *Shape groups* are assembled by partitioning the sixty views into

[3] This system uses the photometric stereo technique to extract shape information; in order to be

classes using the information about the visible surface patches. The interpretation tree contains paths from the root (an object with no classified patches) through intermediate nodes which correspond to a partial match between some model patches and some image patches, to a leaf node which represents a consistent interpretation of a particular view.

Objects in an unknown position are classified and their pose extracted using a two-phase approach. Features are computed from the sensed data (example features are inertia, shape, EGI, and curvature measures). Intermediate classifications are performed at each nonterminal node of the interpretation tree using one of these features. Features for the sensed object are compared with the same features calculated from the CAD model, and the most discriminatory feature (selected by hand) is used at each decision point. Once a terminal node has been reached, the object and its rough pose have been identified. The remaining processing determines the pose of the object using additional feature comparisons that constrain the three rotational degrees of freedom between the sensor and the sensed object. Examples of this technique are presented for several scenes, employing registered intensity, orientation (needle map), and derived range data. The technique seemed accurate in determining the pose of an object (no numerical figures were reported), but its power in discriminating between different objects was not demonstrated.

Bolles *et al.* [15, 16] developed an industrial part recognition system known as 3DPO (3-Dimensional Part Orientation) for identifying and locating parts in a bin-picking application: several copies of a single part in a box are to be located and presumably removed by a robotic arm. The CAD model of the part contains volumetric, surface, and edge information, as well as lists of adjacent object features like bounding edges and holes. Models are comprised of planar and cylindrical patches bounded by circular and linear edges. In addition to this information, the model also contains lists of feature types to be used in matching (like straight and curved edges), accompanied by instances of those feature types in the object. A polyhedral approximation of the object is also stored in the model, and is used primarily to generate range images from hypotheses of identity and pose.

The recognition paradigm (as in Ikeuchi's work [62]) is search-based. Here, the

visible to the sensor, a surface must be illuminated by three light sources. Those surfaces which are shadowed during illumination by one or more of the sources are not detectable by the stereo system, although they may be visible to the intensity sensor.

search process is constrained by restricting the initial matching to just a small number of features. Initial match hypotheses are based on edges. Edges with similar properties are grouped into classes and independent strategies are pursued for each class (these strategies are constructed by hand). By matching on only one feature and verifying the hypothesized match with additional features, considerable pruning of the search space is obtained. The authors report that as few as three features are typically needed to verify a correct match hypothesis. They demonstrate some impressive results on real scenes, although they only worked with a single object model.

Hansen and Henderson [51] make a strong case for the integration of CAD and computer vision, and outline a recognition system employing a *strategy tree.* Object models in this system are constructed using **Alpha_1**, a solid modeler developed at the University of Utah. A variety of features are extracted from the CAD models: pointwise normals, curvatures, and surface classifications (employing curvatures), planar region approximations, edges and arcs, and apparent symmetries (for surfaces of revolution). An important contribution of this work is a systematic approach to the selection of features for recognition. *Feature filters* are employed to select the most powerful (discriminatory) features to use in recognizing a specific object, and discard the others. For example, surface curvatures may be powerful features for recognition of sculpted surfaces, but are essentially useless in recognition of polyhedra. A properly constructed feature filter would reject curvatures when compiling a recognition tree for the polyhedron, and include them in the strategy tree for the sculpted object.

A strategy tree is a generalization of the tree classifier. One such tree is generated for each object in the database. Recognition of an object and computation of its pose corresponds to a descent of the strategy tree for that object. The strategy tree is constructed by placing the most discriminatory features at the first level below the root. Subtrees under the first level are known as *corroborating evidence subtrees* (CESs), which direct a search for evidence that the first-level features are supported, and attempt to refine the pose estimate. If a first-level feature is detected and supported by the CES, a hypothesis is generated about the object's identity and pose. Verification of the generated hypothesis can be done either using frame correlation between the sensed image and a synthesized range image of the model, or through matching of features predicted from the model and the hypothesis with image features. This system was demonstrated only on a single polyhedral object. Its ability to discriminate between different objects and its performance with curved objects were not demonstrated.

Kak *et al.* [71] propose the use of *sensor-tuned representations* as an intermediary between a CSG-based solid modeler (the PADL-2 system [24]) and sensor data (range imagery). Their intermediate representation is an attributed graph in which nodes represent object surfaces (with attributes such as surface type and surface area), and edges represent inter-surface relationships (with attributes such as edge type, and edge angle). The generation of such an attributed graph from a CSG model is relatively straightforward as long as the modeler maintains a surface-based model in parallel with the CSG model (which many do). The attributed graph construction is then mainly a process of assembling the adjacency information not *explicitly* present in the parallel representation and inserting it in appropriate slots in the model graph.

Range images in Kak *et al.*'s system are analyzed as follows. Local operators are used to estimate surface normals and curvatures at each surface point. Jump edges are extracted by thresholding on local changes in depth. Crease edges are obtained by thresholding on the surface curvatures, and are labeled as convex or concave. After extraction and labeling of the three types of edges, the remaining range points are grown into connected regions. Histograms of surface curvatures provide a label for each region from one of the following: planar, generalized cylindrical, generalized spherical, and toroidal. Inter-surface relations are constructed by forming an adjacency matrix for all possible pairs of patches and estimating the orientation change between adjacent patches.

The recognition paradigm used in Kak's work is graph matching. Given the attributed graphs derived from objects constructed in the solid modeler, and another graph derived from the sensed image, the task is to find a graph-subgraph isomorphism between the scene subgraph and one of the model graphs. The combinatorics of this matching problem can be burdensome if the model database is large. The recognition system is discussed in detail and demonstrated on a few simple scenes.

Chen and Kak [27] developed 3D-POLY, perhaps the fastest 3D object recognition vision system described in the literature. Object models are constructed by merging multiple views; primitive surfaces are restricted to be planar or quadric, and matching employs geometric feature sets similar to those used in our system (orientation and location parameters). Surface and edge features are organized into sets which allow a pose estimate to be obtained when a model set is associated with a scene set[4]. Appropriate selection of such sets drastically restricts the number of possible interpretations of the scene primitives. Once a transformation (pose estimate) has been obtained it is verified by comparing the orientation of appropriate transformed model features and associated scene features (such orientations should be similar). These heuristics allowed the complexity of the recognition process to be reduced. 3D-POLY was tested on two scenes containing instances of two different models. Recognition times for the two scenes were 9 and 4 seconds (on a Sun 3 computer); note, however, that the input to the system is a processed range image. The acquisition of the range data and initial segmentation can be time-consuming.

Grimson and Lozano-Pérez [47] developed a system which identifies and estimates the pose of polyhedral objects from sparse range or tactile data. The input to this system is a set of surface points (assumed to be accurately measured) and estimated surface normals at those points (which can be noisy). A correct match in their system corresponds to a path in the interpretation tree, which enumerates all possible assignments of sensed surface points to model surfaces. Two procedures are used to effectively prune the interpretation tree, avoiding the combinatorial problems inherent in a brute-force search. First, four constraints between surface points are used to restrict the search. These constraints embody allowable distances between two points assigned to two model surfaces, the tolerance in orientation difference between the normals at the two points and their associated surfaces, tolerance in projection of a vector joining two surface points on their surface normal, and the correct sign of the triple product of three surface normals at sensed points with respect to the corresponding sign derived from the model. Only assignments of point pairs to model surface pairs which satisfy all four constraints are allowed.

After interpretation tree pruning based upon the local constraints mentioned above, several possible interpretations will often survive, each corresponding to an object identification and pose estimate which was admissible with respect to the *local* constraints. These interpretations are then verified by computing the transformation required to map the model corresponding to the interpretation into the sensed data, and then rejecting interpretations in which the sensed data points fall outside the bounds of their corresponding model faces. If multiple interpretations survive this verification process, they usually reflect object symmetries or artifacts of the

[4] This idea is similar in spirit to the local feature set employed in 3DPO [15].

model-construction process, and are all returned as valid interpretations. Simulation experiments with synthetic data (generated from the models) showed that this method effectively prunes the search space[5]. High-resolution range data from a triangulation sensor was also used as input to the system. Two range images of a polyhedron were processed to extract points lying on planar faces, producing nine points on one object and 11 on the other. For the 9-point image, pruning and transformation computation produced three plausible interpretations. The 11-point image produced two plausible interpretations.

Hoffman *et al.* [55, 57, 67] developed a complete system for 3D object recognition from range data. Object models were constructed on a custom CAD system, and model features used were:

- global features: object silhouette perimeter, number of connected background components, and the number of such components within the convex hull of the object silhouette.
- intra-patch features: a trinary label indicating planarity, convexity, or concavity, surface area, elongation, and a boundary fit measure.
- inter-patch features: edge type (none, jump, adjacent), normal angle, minimum and maximum distances, boundary angle, a boundary fit measure, and the size of the jump between the two patches.

Objects in the scene are recognized as follows. The unknown view is segmented using local feature clustering (see Chapter 4). The initial segments are classified, producing the same features as those listed above. The combination of the initial segmentation with the feature list is referred to as the *initial representation.* Then, object model information is used to perform *knowledge-based merging* of patches in the initial representation, producing a *modified representation* of the scene specific to each model. These modified representations are then input to the recognition module, which contains codified *evidence values* for each object model. The similarity between the features in the model database and each of the modified representations is evaluated, and if one model produces a high similarity value, the unknown object in the scene is identified as that model. A method for generating the codified evidence rules was

[5] In experiments on simulated data, only a few interpretations survived the pruning process, out of a total population greater than 10^{19}.

also developed. This system gave impressive results (less then 1% misclassification rate) using 290 test views drawn from a ten object database.

Vemuri and Aggarwal [118] developed an object localization system employing range data as the source of both object models and input images. Object models are constructed by placing the object of interest on a turntable and merging the 3D images for a number of views (produced by rotating the table) at the range pixel level (interframe registration of the range views is estimated from the position of a line drawn on the turntable). Smooth surfaces are obtained from the set of points using qualitative classifications from surface curvature estimators (see Section 4.1.2). When the system is presented with an input image, curvature features are estimated and regions are classified as above. A candidate set of 'likely' models is selected by examining the similarity between the segmentations of the image and those of the models. The localization procedure is then applied to each model, and an error measure is calculated for each candidate model. The model producing the smallest error is accepted as the true identity.

Localization is attacked as a combination of a single-point correspondence problem (which finds the translation component of pose) and an alignment problem (which fixes the rotational component of pose). Briefly, range pixels in the input image and the model are candidates for correspondence if they lie on lines of identical constant principal curvatures. The search procedure used to extract such candidates is not described in detail. Once such candidates have been identified, a rigid rotation which aligns the Darboux frames (the coordinate systems defined by the surface normal and the two principal curvature directions) on the surface at the given points is computed. A few test images were generated from models constructed from real range views, and errors in the rotational component of the estimated pose ranged from two to five degrees for objects without rotational symmetry.

Fan *et al.* [37] have also developed a 3D vision system employing multiple views as a model source. Unlike Vemuri and Aggarwal, an input image is matched against individual images of the database objects rather than one model constructed from those images. The same processing is performed on range images forming the database and the unknown views to be classified. Connected patches are extracted using the edge-based technique described in Section 4.1.2. Each patch is described in terms of orientation, curvature, occlusion, and location parameters. Heuristics are used to group patches in the image (which may come from more than one object) into classes corresponding to single objects, and patches within each class are linked with attributed relations summarizing the type of adjacency and a probability of connection. The output of the image analysis phase is a set of graphs describing scene objects. Given an input image and its output graphs, a screening module selects database views judged 'likely' to match the input image graph. Construction of associations between entities in a single model image and the scene is then performed, and estimates of object pose are produced concurrently. An analyzer module attempts to cope with segmentation or interpretation errors by modifying the graph structure. The database consisted of 32 views of ten different objects. This system was tested on three scenes containing occlusion and multiple objects, and correct interpretations were obtained for all objects in these scenes.

In his thesis, Chen [28] proposed a search-based 3D object recognition system. The underlying representation of object models and sensed images uses *wing features*[6] rather than specific surface types such as planes and quadrics. Given a set of scene wings and several models described as wing sets (arranged by viewpoint), recognition involves the following steps:

- *Indexing*: the selection of candidate models and views of those models. Models are rejected from consideration if the sets of scene wings and model wings do not intersect. Models surviving this test are arranged in a priority queue, using a measure of similarity between the sensed wing set and the model wing set as the priority function.

- *Consistent Labeling*: The scene wing set is matched to each view of the candidate model set using the interpretation tree search technique.

- *Parameter Estimation*: Hypotheses obtained from the search process produce several pose estimates.

- *Decision*: The 'best' match is detected by detecting clusters in the parameter space of transformations obtained in the estimation step.

The parameter estimation and decision modules draw on earlier work by Stockman [108, 109]. Pose estimates are produced for each correspondence between a scene wing and a model wing. If we view the ensemble of estimates produced from

[6] See Section 2.3.2 for a brief description of wing features.

all hypotheses as a cloud of points in the pose parameter space, we would expect a compact cluster to be present at the correct pose value. Detection of these clusters is addressed in [108] and [28].

Lowe [78] and Huttenlocher and Ullman [61] have also examined 2D-3D matching. Lowe uses the principles of perceptual grouping to identify and rank intermediate-level structures from nonaccidental relationships (proximity, collinearity, and parallelism) between tokens (line segments) in the image plane. Given these segment groupings and a model of the object to be located, an iterative procedure attempts to 'align' corresponding structures, or (equivalently) find the rigid model transformation which, when projected to the image plane, produced the observed image. Huttenlocher and Ullman developed a non-iterative sequential procedure for aligning corresponding point triplets obtained from the model and the scene. Candidate image points used to estimate the transformation are extracted from a multiscale representation.

5.3 Interpretation Tree Search for 3D Recognition

Our computational model for 3D object recognition and localization is heuristic search of the interpretation tree. The IT embodies all possible sequences of *associations* (matchings) between a model surface and a scene surface. Descent in the tree implies an increasing level of commitment to a particular hypothesis of identity and pose for a *single* model object. If the system decides (based on the scene content) that scene patches could match more than one of the database objects, a sequence of IT searches is performed, one for each object[7]. Figure 5.1 illustrated the concept of the interpretation tree. We now develop the formalism.

5.3.1 Notation and Definitions

The input to the model-based 3D vision system is a set of s scene surfaces (or *scene entities*):

$$\mathcal{S} = \{S_1, S_2, \ldots, S_s\}$$

[7] Section 5.3.7 describes our techniques for candidate model selection.

The model under consideration contains m surfaces (or *model entities*):

$$\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$$

An *interpretation* (or a *hypothesis*) is a triple containing the *identity* of the model object under consideration, an ordered set of *associations* between scene and model surfaces, and a rigid *transformation* which aligns the model with the scene entities involved in the interpretation.

$$\mathcal{I} = < \mathcal{O}, \mathcal{A}, \mathcal{T} > = < \texttt{object}, \{(S_1, M_{i_1}), \ldots (S_j, M_{i_j})\}, (\mathbf{R}, \mathbf{t}) >$$

Several associations are required before the transformation $(\mathbf{R}, \mathbf{t})$ is uniquely determined. A hypothesis with a unique transformation is termed *verifiable.*

The IT is constructed by forming all possible interpretations which contain associations ordered on the scene entities S_i; in other words,

$$\mathcal{I} = < \texttt{object}, \{(S_1, M_1), (S_2, M_4), (S_3, M_9)\}, (\mathbf{R}, \mathbf{t}) >$$

is an element of the IT, but

$$\mathcal{I} = < \texttt{object}, \{(S_1, M_1), (S_3, M_4), (S_2, M_9)\}, (\mathbf{R}, \mathbf{t}) >$$

is not, because the S_i are not interpreted in order. Heuristics for ordering scene entities are discussed in a later section. At the root of the IT, the association set $\mathcal{A}$ is empty. At the first level down from the root, the association set contains a single interpretation for S_1. At the second level, $\mathcal{A}$ contains associations for S_1 and S_2, and so on. We define the *depth* $d(\mathcal{I})$ of an interpretation to be the number of associations:

$$d(\mathcal{I}) = \|\mathcal{A}\|.$$

Alternatively, the depth of an association is its distance from the root node.

5.3.2 NULL Associations

The typical scene to be considered in this thesis contains more than one object. Therefore, the scene entity set $\mathcal{S}$ will contain entities from more than one model. Additionally, segmentation errors can produce scene entities which correspond to *no* model. We reconcile the complexity of the scene entity set with the constraint that we look for one model at a time by allowing the *NULL match*[8]. Define a NULL match as an association of the form

$$(S_i, NULL),$$

and meaning that we mark S_i as having been used (and therefore not subject to subsequent association with a model entity), but that it has no effect on the computation of a pose transformation or on verification of a hypothesis.

We define a *correct* association as one corresponding with the true scene interpretation. We will reserve the terms 'correct' and 'incorrect' for non-NULL associations; a NULL association is considered neither correct nor incorrect.

5.3.3 Ranking Scene Entities

Inherent in a search-based approach to recognition is the sequential consideration of scene entities. Since we desire that search be performed efficiently, the order in which scene entities are selected is important. There are two competing goals to be addressed when we order the scene surfaces. Obviously, we prefer to match 'more reliable' entities early in the search, to avoid taking the NULL branch (since it is considered after all model entities have been tried). The reliability of parameter estimates for scene entities is related to their visible area. On the other hand, it is important that we reduce the number of subtrees to be searched *as close to the root node as possible*; pruning is much more powerful near the root than it is further down in the tree. This goal suggests that we should attempt to employ the *most discriminatory* scene entities near the root node. These two goals can conflict, because the most discriminatory entity is not necessarily the largest patch, and vice versa.

At present, we are using patch area as a criterion for entity ranking prior to search. Associations made at the first level of the tree involve the largest scene entity, level-2 associations involve the next largest scene entity, and so on.

5.3.4 Pruning and Predicates

As we illustrated in the example of Figure 5.1, not all interpretations are valid. Clearly, the presence of an incorrect association implies an incorrect interpretation.

[8] or 'wild-card' match [45].

If we reject an interpretation

$$\mathcal{I}_0 =< \texttt{object}, \mathcal{A}_0, (\mathbf{R}_0, \mathbf{t}_0) >$$

as invalid, all interpretations

$$\mathcal{I}_k =< \texttt{object}, \mathcal{A}_k, (\mathbf{R}_k, \mathbf{t}_k) >$$

with $\mathcal{A}_0 \subset \mathcal{A}_k$ are also invalid and need not be generated (much less checked for validity). This is the essence of *pruning*; we wish to *test* interpretations as they are generated, reject ones which are invalid, and avoid searching the subtrees rooted at the incorrect interpretations.

The specific techniques we use to prune incorrect interpretations are discussed in Section 5.3.8. Our general approach is to apply several *predicates* to each interpretation as it is generated. A *general predicate* $\mathcal{P}$ is a mapping

$$\mathcal{P} : \Im \rightarrow \{\text{TRUE, FALSE}\},$$

where $\Im$ is the set of possible interpretations $\mathcal{I}$. These predicates return TRUE if they are satisfied by the interpretation, and FALSE if the constraints of the predicate are violated and the interpretation and its subtree should be pruned. Since the scene and model entities are surface primitives with orientation, location, and intrinsic parameters (*i.e.* geometric parameters), our predicates are geometric in nature.

Intuition suggests one way to test interpretations with the predicates: simply examine the consistency of the 'newest' association in the hypothesis. Such tests are *unary*, as they only involve one scene entity and at most one model entity. However, unary tests alone often do not provide sufficient pruning power to significantly reduce the size of the IT, primarily because of their locality. *Binary* predicates, which examine all *pairs* of associations involving the 'newest' one, allow us to reject interpretations in which binary relationships between a pair of scene entities do not agree with the relationships between the corresponding pair of model entities.

5.3.5 Heuristic Search Termination

We are now in a position where we could give an algorithm for IT search. Such an algorithm would take as its input the scene description and a model description, build sequences of interpretations, and reject invalid ones. When an interpretation at the bottom of the tree (*i.e.* one which has associations for all scene surfaces, some associations possibly NULL) is generated and accepted by the geometric predicates, we would save that interpretation as a possible solution, and perform a verification step (which we discuss in Section 5.5). The interpretation with the highest measure of validity (if any) would then be accepted, and the scene primitives involved could be removed from consideration. The 'new' scene can then be re-interpreted, and so on.

Examination of the IT search in the context of 3D recognition from geometric features yields additional constraints which we can bring to bear on the interpretation problem. As we shall see in Section 5.4, each successive correct non-NULL association provides constraints on the components of the pose estimate. This fact provides us with a mechanism to *terminate* the search *before* reaching the leaf nodes, *i.e.* before associating all surfaces in the scene with either a model surface or NULL. As soon as we have an unique pose estimate, we can terminate the search process and immediately begin the verification step. As a matter of fact, verification of a correct hypothesis can also suggest interpretations for the remaining scene surfaces. This technique, called *hypothesis extension*, is discussed in Section 5.5.

5.3.6 IT Search Algorithm

Putting together the results of the previous sections, we now propose the algorithm (called SEARCH), the heart of BONSAI. This algorithm will compute a list of plausible hypotheses and their scores. A pseudocode version of this algorithm appears in Figure 5.2. The internal details of the procedures labeled COMPUTE_POSE and VALIDATE are discussed in later sections. The list of verified hypotheses is placed on an output list named PLAUSIBLE_HYPOTHESES.

5.3.7 Model Database Pruning

Previous sections have outlined the interpretation process for a single model. Our database contains twenty different object models, each an attributed relational graph. It is computationally unattractive to apply the IT search algorithm once for each model in the database if we can *infer* that most of the objects *cannot* be present in the scene. This section examines techniques for pruning the model database by examination of the scene content prior to IT search. Just as scene entities and their

```
Algorithm SEARCH($\mathcal{I}$ =< ID, $\mathcal{A}$, (R, t) >)
Input: Model ID with surface set $\mathcal{M}$
Input: Scene surface set $\{S_1, \ldots, S_s\}$
Output: PLAUSIBLE_HYPOTHESES
FOR EACH predicate $\mathcal{P}_j$ DO
  IF NOT $\mathcal{P}_j(\mathcal{I})$ RETURN
ENDFOR
(R, t) ← COMPUTE_POSE($\mathcal{I}$)
IF (R, t) is unique THEN
   score ← VERIFY($\mathcal{I}$)
   Place (score,$\mathcal{I}$) on PLAUSIBLE_HYPOTHESES
   RETURN
ENDIF
$m \leftarrow \|\mathcal{M}\|$
$\mathcal{M}_{\text{unused}} \leftarrow \mathcal{M} - \bigcup_{j=1}^{d}(\mathcal{I})M_{i_j}$
IF ($d(\mathcal{I}) = s$) RETURN
FOR $i = d(\mathcal{I}) + 1$ TO $m$ DO
   $\mathcal{A}' \leftarrow \mathcal{A} \cup \{(S_{d(\mathcal{I})+1}, M_i)\}$
   $\mathcal{I}' \leftarrow <$ ID, $\mathcal{A}'$, {R, t} >
   SEARCH($\mathcal{I}'$)
ENDFOR
$\mathcal{A}' \leftarrow \mathcal{A} \cup \{(S_{d(\mathcal{I})+1}, NULL)\}$
$\mathcal{I}' \leftarrow <$ ID, $\mathcal{A}'$, {R, t} >
SEARCH($\mathcal{I}'$)
END Algorithm SEARCH
```

Figure 5.2: Algorithm SEARCH: Constrained search of the interpretation tree.

relationships constrain interpretation for a single model, they also restrict the set of models to be considered. In this thesis, we employ a few simple techniques for reducing the database of models to examine. However, a detailed off-line examination of the distribution of geometric parameters and relations between model entities can 'flag' certain features as *salient* (*i.e.* highly discriminatory, informative, and powerful). Exploiting this saliency at run-time (when objects are presented to the sensor) is a topic of future research, and could supplement or even partially supplant IT search as a recognition mechanism. Our model database pruning procedure rejects models from the model database if the scene contains features that the models do not. Specifically, we look for curved scene surfaces, and reject models which do not contain surfaces with similar intrinsic parameters. We also examine the distribution of angles between the orientation parameters of patches joined at crease edges, and reject models without similar inter-patch angles.

Intrinsic Properties of Curved Surfaces

The presence of intrinsic shape parameters allows us to avoid the consideration of object models which do not possess surfaces with similar parameters as those in the scene. As part of the process of bringing a new object model into the recognition system, we compute a **radius list**, which contains a sorted list of the radii of curved model surfaces. Since our system utilizes only spherical and cylindrical surfaces, radii are the only intrinsic shape parameters. However, this notion could be generalized to more complicated surface types easily, by defining additional lists (such as a **vertex-angle** list for conical surfaces, a **shape parameter list** for superquadric surfaces, *etc.*).

When a segmented image and its symbolic description is made available to the recognition system, the intrinsic parameters of all curved image surfaces are extracted, and sequentially compared to the radius lists of each model. Those models which have entities of similar radii to the scene entities are considered for IT search, but the remaining models, which do not contain primitives with similar intrinsic parameters, are eliminated from further consideration.

Inter-patch angles

In addition to examination of unary (patch-specific) features like the intrinsic parameters, we can also reject models based on the absence of salient binary relationships between patches that were observed in the scene. We have implemented a model database pruning step which examines the angles between the orientation parameters of primitive surfaces in the scene and the models. As an off-line process, we create a histogram of inter-patch angles (at present we use a 90-bin histogram with each bin being one degree 'wide'). At recognition time, the inter-patch angles for scene surfaces *connected by a crease edge* are compared with the histograms for each model, and models which do not have similar angles are not searched. We restrict our angle measurements to those between connected patches as an attempt at only pruning within scene objects. One can easily imagine multi-object scenes where a salient angle is observed between surface primitives belonging to different objects! The presence of this angle might prevent a model or set of models from being rejected.

These simple model database pruning procedures were designed specifically to be *conservative.* We do not wish to reject the correct model, and the tolerances on intrinsic parameters and angle differences are deliberately loose in order to minimize the chances of such a rejection. One unfortunate byproduct of this conservative approach is that more models must be searched; however, Algorithm SEARCH can also reject models (by producing no valid hypotheses), so the additional investment in computation is preferable to obtaining an incorrect answer.

5.3.8 Geometric Predicates

In this section, we describe the unary and binary predicates used to prune the IT. As outlined in Algorithm SEARCH, these predicates are applied sequentially to each new hypothesis as it is generated (by adding a new association to the existing set). If any one of these predicates is not satisfied, the new interpretation is rejected and its subtree is pruned. This procedure is depicted graphically in Figure 5.3. The pruning power of these predicates changes with the model under consideration and the depth in the tree. It would be interesting to study the effect of ordering on these predicates. If those predicates which provide the most pruning power at the current level are applied first, some increase in speed might be observed, although the improvement would probably be marginal, since the predicates are not computationally

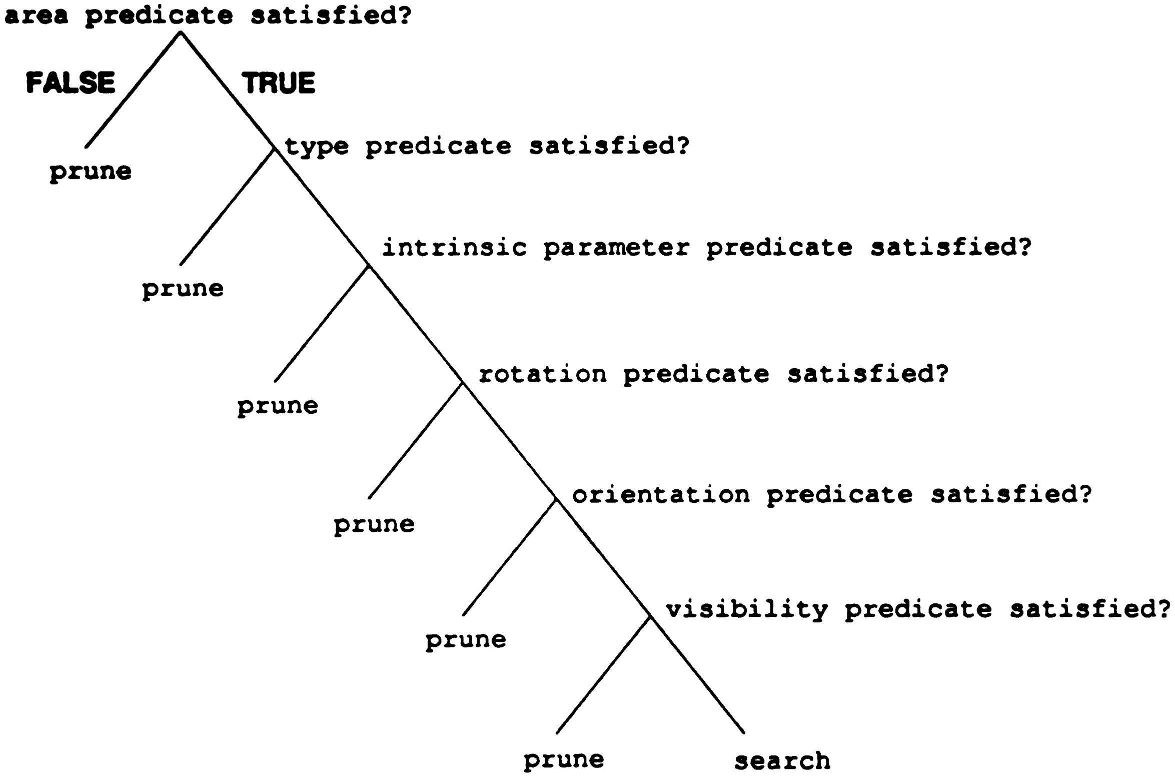
area predicate satisfied?
FALSE
TRUE
prune
type predicate satisfied?
prune
intrinsic parameter predicate satisfied?
prune
rotation predicate satisfied?
prune
orientation predicate satisfied?
prune
visibility predicate satisfied?
prune
search

Figure 5.3: Sequential application of geometric predicates.

demanding. We also note that the predicates are not totally independent from one another; an incorrect interpretation is often rejected by more than one predicate. In such situations, we would hope to use the 'cheaper' predicate before the others, so that rejection of the interpretation is as efficient as possible. These issues are topics for future research.

Unary Predicates

Three unary predicates have been implemented in the current version of BONSAI. All apply a test to the newest association in the interpretation being evaluated. In order to simplify the discussion, we are assuming that the newest association is non-NULL. Because NULL associations should not affect the interpretation process either positively or negatively; these predicates have the value of TRUE when the newest association is NULL.

The first is the *area predicate* $\mathcal{P}_{\text{area}}(\mathcal{I})$, defined as follows. Let the 'newest' association in $\mathcal{I}$ be (S_i, M_{n_i}). Define A_{S_i} to be the estimated area of S_i (calculated during segmentation and classification), and $A_{M_{n_i}}$ to be the *maximum* estimated area of M_{n_i}, calculated from the 320 synthetic views of the model generated as part of the model-building process. Then

$$P_{\text{area}}(\mathcal{I}) = \begin{cases} TRUE & \text{if } A_i \leq A_{M_{n_i}} \\ FALSE & \text{if } A_i > A_{M_{n_i}} \end{cases} .$$

This predicate should appeal to intuition: we reject associations between scene primitives and model primitives for which the area estimated in the scene is larger than the *largest possible* area of the model patch. Empirical performance shows that this predicate is most powerful (*i.e.* it rejects the most interpretations) at the top of the IT, because the first surfaces to be interpreted are the largest and the least likely to be self-occluded or occluded by other objects in the scene[9].

Another unary predicate used by BONSAI is the *type predicate* $\mathcal{P}_{\text{type}}$. Given the interpretation $\mathcal{I}$ with 'newest' association (S_i, M_{n_i}), $\mathcal{P}_{\text{type}}(\mathcal{I})$ is TRUE if S_i and M_{n_i} are the same primitive type (*i.e.* plane, cylinder, sphere).

[9] If we assumed that the scenes to be interpreted contained only *one* object, we could strengthen this predicate by requiring the model and scene areas for an association to agree within some tolerance value for at least one of the 320 model views. As an added benefit, this improved predicate would reject both incorrect models, and incorrect *views* of the correct model.

The third unary predicate is the *intrinsic parameter predicate* $\mathcal{P}_{\text{intrinsic}}$, which is defined only for curved surfaces. This predicate is applied after $\mathcal{P}_{\text{type}}$, so we are assured that the newest association's component surfaces are both spherical or both cylindrical. The intrinsic parameter predicate returns TRUE if the intrinsic parameters (radii) of S_i and M_{n_i} are within a specified tolerance value of one another (our implementation uses an empirically-chosen tolerance of 0.35 inches).

Binary Predicates

BONSAI employs four binary predicates to prune invalid IT subtrees. These predicates examine all *pairs* of associations in the interpretation $\mathcal{I}$ under consideration, and return FALSE if the constraint specific to the predicate is violated for one or more of the pairs. As in the unary case, we will limit our discussion to situations where the newest association is non-NULL. Additionally, we note that if the other association in one of the pairs is NULL, the constraint is not applied, and the final decision of the predicate is not affected. Thus, if the current predicate contained the associations

$$\mathcal{A} = \{(S_1, M_4), (S_2, NULL), (S_3, M_8), (S_4, M_1)\},$$

the conditions of the predicate would be applied to the pairs

$$[(S_1, M_4), (S_4, M_1)], [(S_3, M_8), (S_4, M_1)],$$

but not to $[(S_2, NULL), (S_4, M_1)]$.

In Section 5.4, we will describe our method for estimating the rotational and translational components of the object pose (this *pose transformation* aligns the model with the image). Each pair of non-NULL associations not involving one or more spheres[10] produces an estimate of the model rotation. If these estimates vary widely, the interpretation is incorrect and should be pruned. This idea is the basis for the *rotation validity predicate* $\mathcal{P}_{\text{rotation}}(\mathcal{I})$, which is TRUE if the model rotations estimated for each pair of associations agree within a prespecified tolerance (chosen to be 0.25 radians, or 14 degrees), and FALSE otherwise.

The *orientation predicate* is perhaps the most intuitive binary predicate. As above, association pairs containing one or more spheres are excluded from this predicate since

[10] The rotation is estimated from the patch's orientation parameter, and spheres lack such a parameter, so we cannot estimate rotation from a pair of associations containing a spherical surface.

spheres do not possess orientation parameters. This predicate examines the difference in orientation parameters between the scene primitives in the association pair and the corresponding model primitives. Let $\mathbf{p}_{s1}$ and $\mathbf{p}_{s2}$ be the orientation parameters of the two scene primitives in the association pair, and let $\mathbf{p}_{m1}$ and $\mathbf{p}_{m2}$ be the corresponding model primitive orientation parameters. Then

$$\begin{aligned} \Delta_s &= \cos^{-1}\frac{\mathbf{p}_{s1}\cdot\mathbf{p}_{s2}}{\|\mathbf{p}_{s1}\cdot\mathbf{p}_{s2}\|}, \text{ and} \\ \Delta_m &= \cos^{-1}\frac{\mathbf{p}_{m1}\cdot\mathbf{p}_{m2}}{\|\mathbf{p}_{m1}\cdot\mathbf{p}_{m2}\|}, \end{aligned}$$

and the predicate is satisfied only if $\|\Delta_s - \Delta_m\|$ is less than a threshold (6 degrees in our experiments) *for all association pairs.*

The *visibility predicate* is a binary predicate related to the unary area predicate discussed above. Section 2.4.5 described the construction of area lists for each object model surface, and the generation of *never-simultaneously-visible* relations between surfaces which don't appear together in any view. The visibility predicate is satisfied only if, for all non-NULL association pairs, the model entities are *not* connected by a *never-simultaneously-visible* relation.

5.4 Pose Estimation

Part of BONSAI's output is a pose estimate for each object model that it decides is present in the scene. This section gives our techniques for pose estimation. The input to the pose estimation portion of BONSAI is an interpretation, assumed to have enough non-NULL correspondences to make the pose estimate unique. The output is a 3 × 3 rotation matrix $\mathbf{R}$ and a 3D translation $\mathbf{t}$. The system uses two different techniques to estimate pose; one technique is suitable for 'general' objects whose orientation parameters are not all parallel, and the other technique was developed to get unique rotation and translation estimates for objects with rotational symmetry.

5.4.1 Nonsymmetric Model Pose Estimation

We adopt Grimson and Lozano-Pérez' technique [47] for estimation of model rotation. Their presentation was developed for planar surfaces, but generalizes easily to any surface with an unique orientation parameter. Chen *et al.* [30] extended Grimson and Lozano-Pérez' method to include line segments in addition to planar patches.

They describe algorithms for the cases of: (i) two nonparallel scene lines and two corresponding model lines, (ii) two noncoplanar scene planes and a scene line and corresponding model planes and line, and (iii) three (non-coplanar) scene planes and corresponding model planes. Our technique obtains a rotation estimate for each pair of non-NULL associations not involving spherical patches. The ensemble of estimates can be checked for validity and averaged to produce the final estimate.

We now outline the procedure for a single pair of scene and model surfaces. Let these entities be associated as (S_{i_1}, M_{j_1}) and (S_{i_2}, M_{j_2}), and let the entities have orientation parameters $\mathbf{p}_{s1}$ $\mathbf{p}_{m1}$, $\mathbf{p}_{s2}$, and $\mathbf{p}_{m2}$, respectively. We are interested in a 3×3 rotation matrix which, when applied to $\mathbf{p}_{m1}$, aligns it with $\mathbf{p}_{s1}$, and aligns $\mathbf{p}_{m2}$ with $\mathbf{p}_{s2}$. An equivalent representation for this rigid 3D rotation is composed of an *axis of rotation* (constrained to pass through the origin) $\mathbf{r}$ and an *angle* θ [97, pp. 121-127]. We determine $\mathbf{r}$ and θ as follows [47].

We wish to rotate $\mathbf{p}_{m1}$ into $\mathbf{p}_{s1}$. Rigid rotations through an axis $\mathbf{r}$ must preserve the angles between $\mathbf{r}$ and $\mathbf{p}_{s1}$, and $\mathbf{r}$ and $\mathbf{p}_{m1}$:

$$\mathbf{r} \cdot \mathbf{p}_{s1} = \mathbf{r} \cdot \mathbf{p}_{m1},$$

or

$$\mathbf{r} \cdot (\mathbf{p}_{m1} - \mathbf{p}_{s1}) = 0.$$

Hence, given two *non-parallel* model vectors and their corresponding scene vectors (belonging to associated scene surfaces), the rotation which aligns model vectors with corresponding scene vectors is the vector in the direction

$$(\mathbf{p}_{m1} - \mathbf{p}_{s1}) \times (\mathbf{p}_{m2} - \mathbf{p}_{s2})$$

The ***amount*** of rotation θ is given by

$$\theta = \tan^{-1}\left[\frac{(\mathbf{r} \times \mathbf{p}_{s1}) \cdot \mathbf{p}_{m1}}{1 - (\mathbf{p}_{s1} \cdot \mathbf{p}_{m1})}\right].$$

The range of values for the components of $\mathbf{r}$ and θ are examined for each pair of associations in order to verify that the rotation is valid. The estimated rotation axes and angles for an incorrect interpretation will typically vary widely, and we detect and reject interpretations for which this happens. If the rotation axis and angle estimates vary only slightly (less than 14 degrees in our experiments), we average those estimates to get an average axis $\bar{\mathbf{r}} = (\bar{r}_x, \bar{r}_y, \bar{r}_z)$ and $\bar{\theta}$, which are converted to the 3×3 rotation matrix using the following formula:

$$\begin{aligned}\mathbf{R} &= \cos\bar{\theta}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} + \left(1-\cos\bar{\theta}\right)\begin{bmatrix}\bar{r}_x^2 & \bar{r}_x\bar{r}_y & \bar{r}_x\bar{r}_z\\ \bar{r}_y\bar{r}_x & \bar{r}_y^2 & \bar{r}_y\bar{r}_z\\ \bar{r}_z\bar{r}_x & \bar{r}_z\bar{r}_y & \bar{r}_z^2\end{bmatrix}\\ &\quad + \sin\bar{\theta}\begin{bmatrix}0 & -\bar{r}_z & \bar{r}_y\\ \bar{r}_z & 0 & -\bar{r}_x\\ -\bar{r}_y & \bar{r}_x & 0\end{bmatrix}.\end{aligned}$$

After the rotation is estimated, the model and scene surfaces will have similar orientations, but will not in general be coincident in 3D space. Estimation of the translation necessary to align them is the next step. Our approach to translation estimation is (to our knowledge) unique. Its two primary advantages are the following.

- *Linearity*: The translation estimate is obtained from a solution to a linear least-squares problem.
- *Unification*: The least-squares problem contains information from *all* the surfaces involved in the interpretation under consideration.

The primary difficulty in this estimation procedure is accommodation of three different surface types, each of which provides a different number of constraints on the translation's three components. Spheres constrain all three coordinates of the translation estimate. Cylinders constrain two of the coordinates (leaving translation *along the axis* unconstrained). Planes only constrain one coordinate (in the direction normal to the plane), leaving two free translations. How do we put these different constraints together into a linear least-squares framework? For an interpretation which contains associations involving j spherical surfaces, k planes, and l cylinders, we can set up a linear system of $3j + k + 3l$ equations in $3 + l$ unknowns $(\Delta x, \Delta y, \Delta z, t_1, \ldots t_l)$, as follows.

Spheres

Let (x'_m, y'_m, z'_m) be the *rotated* location parameter of the spherical model surface (*i.e.* the center of the model sphere after transformation by the matrix $\mathbf{R}$ estimated above). Let (x_s, y_s, z_s) be the location parameter of the corresponding scene sphere. Clearly, then, we can obtain three equations in the three unknowns $(\Delta x, \Delta y, \Delta z)$:

$$\begin{aligned} x_s &= x'_m + \Delta x \\ y_s &= y'_m + \Delta y \\ z_s &= z'_m + \Delta z. \end{aligned} \tag{5.1}$$

Planes

If the rotation is correct, the rotated model plane and the corresponding scene plane will have identical orientation parameters, which we denote (a, b, c). Their location parameters will in general be different; we denote them d_m and d_s, for the transformed model location parameter and the scene location parameter, respectively. The rotated model plane has the implicit equation

$$ax'_m + by'_m + cz'_m + d_m = 0. \tag{5.2}$$

and the scene plane has implicit equation

$$ax_s + by_s + cz_s + d_s = 0, \tag{5.3}$$

where (x_s, y_s, z_s) and (x'_m, y'_m, z'_m) are the scene and *rotated* model coordinate systems, as in Equation 5.1. But we have assumed that the rotation is correct, and that only a translation is required to bring the two planes into alignment. So, by substituting Equation (5.1) into Equation (5.3) and subtracting the result from Equation (5.2), we have a single linear constraint on the translation $(\Delta x, \Delta y, \Delta z)$:

$$a\Delta x + b\Delta y + c\Delta z + (d_s - d_m) = 0.$$

Cylinders

The case for cylindrical surfaces is slightly more complicated. Recall that the orientation parameter for a cylinder is a vector parallel to its axis, and the location parameter is a point on the axis. Assuming again that the estimated rotation is correct, application of $\mathbf{R}$ to the model orientation parameter will orient the model cylinder properly but not (in general) make it coincident with the corresponding scene cylinder. Let (x'_m, y'_m, z'_m) represent the rotated location parameter of the model cylinder, and let (v_x, v_y, v_z) be the coordinates of the orientation parameter of the scene cylinder (parallel to the rotated model cylinder). We wish to find the translation $(\Delta x, \Delta y, \Delta z)$ which moves (x'_m, y'_m, z'_m) to the *corresponding* point on the axis of the scene primitive. In effect, we have transformed the cylinder-matching problem to a line-matching problem.

Figure 5.4 illustrates the idea behind our solution. We can decompose the overall translation into a component parallel to (v_x, v_y, v_z), and a component perpendicular to (v_x, v_y, v_z):

$$(\Delta x, \Delta y, \Delta z) = (\Delta x_{\|} + \Delta x_{\perp}, \Delta y_{\|} + \Delta y_{\perp}, \Delta z_{\|} + \Delta z_{\perp}). \tag{5.4}$$

We also know that for some number (an *auxiliary parameter*) t,

$$(\Delta x_{\|}, \Delta y_{\|}, \Delta z_{\|}) = t(v_x, v_y, v_z). \tag{5.5}$$

We can derive $(\Delta x_{\perp}, \Delta y_{\perp}, \Delta z_{\perp})$ from (x'_m, y'_m, z'_m), (v_x, v_y, v_z), and (x_s, y_s, z_s). Putting together Equations (5.4) and (5.5), we obtain three equations in $(\Delta x, \Delta y, \Delta z)$ and the auxiliary parameter t:

$$\begin{bmatrix} 1 & 0 & 0 & -v_x \\ 0 & 1 & 0 & -v_y \\ 0 & 0 & 1 & -v_z \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \\ t \end{bmatrix} = \begin{bmatrix} \Delta x_{\perp} \\ \Delta y_{\perp} \\ \Delta z_{\perp} \end{bmatrix}.$$

Note that each cylinder requires its own auxiliary parameter t.

5.4.2 Rotationally-Symmetric Objects

The technique for rotation estimation given above does not work when all of the model surfaces' orientation parameters are parallel (or antiparallel). Objects with rotational symmetry are common among manufactured parts, so we developed an alternative approach to rotation and translation estimation which works with such objects. The estimation of rotation actually becomes simpler for such cases, as one of the three components of rotation is immaterial (rotation about the symmetry axis); there are an infinity of 3 × 3 rotation matrices which can be used as the 'correct'

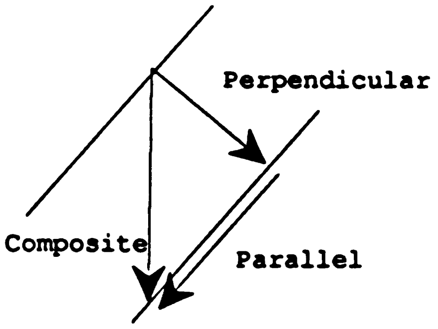

Figure 5.4: Translation estimation: aligning cylinder axes.

orientation transform. BONSAI assumes that models with rotational symmetry are 'flagged' as such; also, the direction of the axis of symmetry (in model coordinates) must be known. Since a mechanical designer typically knows when he/she is designing a symmetric part, and also knows the symmetry axis, these quantities need not be inferred by a computer program.

Consider an interpretation $\mathcal{I}$ containing several associations

$$[(S_1, M_{i_1}), (S_2, M_{i_2}), \ldots, (S_k, M_{i_k})].$$

Ignoring NULL associations and associations involving spherical patches, let the orientation parameters of the entities in the remaining associations be given by

$$(\mathbf{p}_{s1}, \mathbf{p}_{m1}), \ldots (\mathbf{p}_{sk}, \mathbf{p}_{mk}).$$

The condition of rotational symmetry requires that all the $\mathbf{p}_{mj}$ be parallel (or antiparallel). If the interpretation $\mathcal{I}$ is correct, the orientation parameters of the scene surfaces will also be parallel. Let

$$\hat{\mathbf{p}} = \frac{\sum_{i=1}^{k} \mathbf{p}_{si}}{\left\|\sum_{i=1}^{k} \mathbf{p}_{si}\right\|}$$

be the average of the associations' scene orientation vectors. The rotation axis $\mathbf{r}$ and angle θ needed to orient the model so that its orientation parameters are parallel to the observed scene vectors are given by

$$\begin{aligned} \mathbf{r} &= \hat{\mathbf{p}} \times \mathbf{p}_{m1} \\ \theta &= \cos^{-1} \mathbf{p}_{m1} \cdot \hat{\mathbf{p}}. \end{aligned}$$

Once the rotation has been found, we can easily estimate the translation. An estimate of the three components $(\Delta x, \Delta y, \Delta z)$ of the translation vector is produced from each association between planes or spheres. For the planar case, the intersection of the axis of symmetry and the plane under consideration is uniquely determined, so the displacement between the (rotated) model plane's intersection point and the scene's intersection point provides an estimate. For spheres, the displacement between the (rotated) model sphere's center and the scene sphere's center is an estimate of the translation. The average of these values is taken as the final translation estimate.

5.5 Hypothesis Verification and Extension

Ideally, we would like BONSAI to discard all incorrect interpretations that it generates, with at most one (the correct interpretation) surviving. Our use of tolerances on pruning predicates, combined with sensor noise, segmentation errors, and inherent ambiguities among object models, make a hypothesis *verification* step necessary. Each hypothesis surviving the IT search is verified by synthesizing a range image and a segmentation of the object from the faceted approximation to the geometric model described in Chapter 2, and an area-based matching score is calculated by pixel-to-pixel comparison with the input range image and segmentation. Correct interpretations will have the highest matching score.

The matching score for a verifiable interpretation $\mathcal{I}$ is given by

$$s(\mathcal{I}) = s_l(\mathcal{I}) s_r(\mathcal{I}),$$

where $s_l(\mathcal{I})$ is a score computed from the synthetic segmentation and the input scene segmentation, and $s_r(\mathcal{I})$ is a score computed from the synthetic and input range images. The range image score $s_r(\mathcal{I})$ is the fraction

$$\frac{N_+}{N_+ + N_-},$$

where N_+ is called the *positive evidence count*, which we define simply as the number of times that corresponding range pixels in the synthetic and real images have z-coordinates 'close' to one another ('close' is defined as 0.1 inch in our experiments). N_- is the *negative evidence count*, defined as the number of times in which the synthetic image pixel is 'higher' (*i.e.* has a greater z-coordinate) than the corresponding pixel in the observed range image[11]. Although it does not enter into the calculation of $s_r(\mathcal{I})$, a third situation is possible: the observed range pixel could be 'higher' than the corresponding pixel in the synthetic range image. Occurrences of this sort should not (by themselves) be interpreted as negative evidence for the interpretation, because the input image could be reflecting scene occlusion. However, if we assumed that scene objects are isolated, *any* deviation in either direction between the input and synthetic image pixels would tend to indicate that the interpretation is incorrect.

The labeling score $s_l(\mathcal{I})$ is slightly more complicated, but the information produced by comparing the scene and synthetic labelings has a number of benefits in terms of interpretation refinement. Given the two label image, we compute the following quantities.

- n_{S_i,M_j}, the number of places where corresponding pixels in the images belong to scene entity i and model entity j, for all i, j. This could be called an 'overlap list'; ideally, we want n_{S_i,M_j} to be large for (S_i, M_j) to be an element of the associations of $\mathcal{I}$ or for correct associations that aren't present in $\mathcal{I}$ because of search termination.

- N_{good}, the sum of those n_{S_i,M_j} which reflect correct associations in $\mathcal{I}$.

- N_i, the number of labeled pixels in the input image.

- N_s, the number of labeled pixels in the synthetic label image.

- N_{bad1}, the number of image locations where the input label was valid but the synthetic label was not.

- N_{bad2}, the number of image locations where the synthetic label was valid but the input label was not.

Note that the 'bad' quantities only reflect segmentation errors involving unlabeled (background) pixels. *We ignore incorrect labelings involving non-background pixels*, because they could be due to occlusion of the model under consideration by other objects. As above, if we could guarantee that objects would be isolated in the scene, we could incorporate such incorrect labelings. $s_l(\mathcal{I})$ is given by

$$\frac{N_{good}}{N_i}\left[1-\frac{N_{bad1}}{N_i}\right]\left[1-\frac{0.5N_{bad2}}{N_i}\right].$$

[11] Similar measures were used by Hansen and Henderson [51] and Bolles and Horaud (in 3DPO) [15].

By inspection we can see that $0 \leq s_l(\mathcal{I}) \leq 1$. To attain values close to 1, the following conditions are necessary.

- The overlap for correct associations in $\mathcal{I}$ must be high.
- The 'background' portions of the synthetic and input labelings must substantially overlap.
- Most of the correct associations must be present in $\mathcal{I}$.

The third condition is of greatest importance. As mentioned earlier, one of the advantages of BONSAI is its ability to *stop searching* when a verifiable interpretation

$$\mathcal{I}_v = < \mathcal{O}, \mathcal{A}_v, (\mathbf{R}_v, \mathbf{t}_v) >$$

is obtained (typically after three non-NULL associations are present). If the maximal correct interpretation

$$\mathcal{I}_m = < \mathcal{O}, \mathcal{A}_m, (\mathbf{R}_m, \mathbf{t}_m) >$$

is such that $\mathcal{A}_v \subset \mathcal{A}_m$, then we should give $\mathcal{I}_v$ a large score. However, the absence of the associations in $\mathcal{A}_m - \mathcal{A}_v$ will depress the label-based matching score $s_l(\mathcal{I}_v)$. Experiments show that the scores will still be larger than those of incorrect interpretations (because the effects of early search termination are also present in incorrect interpretations), but the adjustment of matching scores for these cases is an interesting idea to pursue in the future.

The presence of the overlap populations N_{S_i,M_j} can suggest additional associations to be added to verifiable interpretations. We add such associations as follows. For each un-associated model surface that has a nonzero population in the synthetic labeling, we find all scene entities which overlap it. If one of those scene entities is also unassociated or associated with NULL, and the amount of overlap is greater than half the population of the model entity in the synthetic labeling, we add the new association containing the scene and model entities to the interpretation. If this process produces any change in the interpretation's association, we then immediately recalculate the matching score, which will typically increase for correct interpretations.

5.6 Analysis of Interpretation Tree Search

In this section, we introduce an error model for the probabilistic analysis of interpretation tree search. Our model is a generalization of the model introduced by Grimson [45]; he employed it to obtain closed-form bounds for the number of nodes expanded during the search process. This section explicitly describes the difference between Grimson's interpretation trees and those used in this thesis. Our IT model, while probably more appropriate for recognition using complex scene entities, does not admit closed-form bounds for the number of nodes visited during search. In the absence of formal techniques, we describe a simulation procedure for obtaining these bounds empirically, and test these bounds against actual numbers obtained from a few recognition experiments.

5.6.1 IT Search as Sequential Hypothesis Testing

We can view the unary and binary predicates developed in Section 5.3.8 in a more formal way by drawing an analogy with traditional statistical inference. In applying a predicate to an interpretation, we are in effect deciding on the validity of a model for the scene. Arbitrarily, we choose the model corresponding to *correctness* (of the interpretation) as a 'null hypothesis' H_0, which we test with the geometric predicates. If the predicate's constraints are violated, we are in effect 'rejecting' this null hypothesis in favor of a general alternative H_a.

Once we have put the application of predicates in such a framework, we can immediately discuss the possible errors in decision-making, and impose a simple model for these errors. Single tests in traditional inference admit two errors:

- A *Type I error* (or *false reject*) occurs if we reject the null hypothesis when in fact it is correct, *i.e.* our predicates reject an interpretation whose associations are all correct.

- A *Type II error* (or *false accept*) occurs if we accept H_0 when in fact H_a applies, *i.e.* our predicates fail to detect an incorrect association and the subtree of the interpretation under question is searched instead of being pruned.

We are adopting terms from statistical inference because of their intuitive similarity to the particular geometric testing situation at hand, *not* because we believe that a simple statistical 'model' applies to any of the test quantities examined when a predicate is evaluated. Continuing the analogy, IT search can be viewed as sequential hypothesis testing. The tests involved examine continuous (*e.g.* radii), and nominal-valued (*e.g.* patch type) data describing the scene and the models. In the traditional situation, thresholds on appropriate test statistics and confidence intervals can be obtained from theoretically-determined distributions; in our situation, however, the mix of data types, the difficulty of modeling the error process, and the complexity of parameter estimation make theoretical thresholds prohibitive (or impossible) to obtain. Thresholds are chosen empirically, appealing to intuition where possible.

In the interests of simplicity, we will model the *ensemble* of geometric predicates (unary and binary) with two hypothesis tests: one unary and one binary. We assume that these predicates are applied sequentially to each interpretation generated during search, and generalize an error model developed by Grimson [45, 46] to allow both Type I and Type II errors. We will assume constant error probabilities and examine (via simulation) growth in the interpretation tree as a function of model and scene size (in primitives), and the number of correct associations made. A more realistic exploration would use families of error probabilities specific to

- the particular test involved,
- the object model involved, and possibly
- the level in the IT being explored.

While this generalization is amenable to simulation, formal bounds on the amount of search have proved resistant to analysis.

5.6.2 Unary Test

In applying our 'generic' unary predicate to an interpretation $\mathcal{I}$, we encounter four alternatives in terms of the model entity under consideration and the correctness of the association involving it. Our unary predicates were designed (see Section 5.3.8) to (in current terms) *accept* the null hypothesis if the newest association is NULL. If the newest association is non-NULL, our test is applied and produces a decision to accept or reject the null hypothesis of correctness. Assume that the Type II error probability is p_1, and that the probability of a Type I error is $1 - p_3$ (we use p_3

so that our notation agrees as much as possible with Grimson's [46]). In summary, the ***probability of acceptance*** or *consistency* [46]) in terms of the newest association (S_i, M_j) in the interpretation under consideration is given by

$$q_{(S_i,M_j)} = \begin{cases} 1 & \text{if } M_j \text{ is null.} \\ p_1 & \text{if } (S_i, M_j) \text{ is an incorrect association } (H_a \text{ applies)} \\ & \text{and } M_j \text{ is non-NULL} \\ p_3 & \text{if } (S_i, M_j) \text{ is a correct association } (H_0 \text{ applies)} \\ & \text{and } M_j \text{ is non-NULL} \end{cases}$$

In Grimson's work, $p_3 = 1$, in effect stating that false rejections are impossible. By allowing $p_3 \neq 1$, we can no longer guarantee that any particular correct interpretation will survive and be verified. The value of p_3 depends on the noise present in the data, the segmentation procedure employed (particularly the sorts of errors it is prone to make), and scene effects such as accidental alignment or occlusion. In practice, we would like p_3 to be as close to 1 as possible while avoiding a large number of false acceptances.

5.6.3 Binary Test

We also need to model the consistency of an arbitrary pair of associations. Let $q_{(S_i,M_j);(S_k,M_l)}$ denote the consistency probability of a pair of associations (S_i, M_j) and (S_k, M_l). As above, this probability should have a discrete distribution depending on the types of surfaces involved and the binary pruning predicates. For simplicity, we define a similar two-type error model, so that

$$q_{(S_i,M_j);(S_k,M_l)} = \begin{cases} 1 & \text{if } M_j \text{ or } M_l \text{ or both are NULL.} \\ p_2 & \text{if } (S_i, M_j) \text{ and } (S_k, M_l) \text{ are not correct} \\ & \text{and } M_j \text{ and } M_l \text{ are both non-NULL.} \\ p_4 & \text{if } (S_i, M_j), (S_k, M_l) \text{ is correct and} \\ & M_j \text{ and } M_l \text{ are both non-NULL.} \end{cases}$$

In Grimson's analysis, $p_4 = 1$, implying that the binary predicate cannot make a type I error. As with p_3, p_4 depends on noise, segmentation quality, and the types of scenes expected; we would want p_4 to be close to 1.

5.6.4 Learning Error Probabilities

In section 5.6.6, we specify the error probabilities using intuition. However, it is possible to estimate these quantities by examining the object models and the behavior of the tests when applied to scenes. Consider the binary *visibility* predicate defined on two patches, which is TRUE if they are simultaneously visible in one of the synthetic views generated during model-building, and FALSE if they are never simultaneously visible. The false-acceptance probability p_2 could be estimated as

$$\frac{N_{true}}{\binom{n}{2}},$$

where n is the number of surfaces in the model, and N_{true} is the number of times (out of the $\binom{n}{2}$ choices of surface pairs) that the members of the pair are simultaneously visible. Similar estimators for other binary predicates could be developed. Learning the error probabilities p_1 and p_3 for unary tests requires, in addition to the model information, an error mechanism for the segmentation process. If we consider the area predicate (in which we reject hypotheses for which the area of the scene and surface in the newest association is greater than the maximum derived areas of the corresponding model surface), the false-reject probability depends on our mechanism for estimating areas, which in turn depends on the segmentation process. Due to occlusion, the area predicate becomes less powerful for rejecting false hypotheses as we descend in the IT. Therefore, the probabilities would depend on level as well as the specific model involved. Instead of estimating the p_i by examination of the model database, we could simply perform Monte Carlo trials on known scenes and count the number and types of errors for a variety of scene and model types, and noise levels.

5.6.5 Search Tree Size

In its traditional incarnation [45], model entities are 're-usable' during IT search. When a given node in the tree is expanded, we can form new associations involving *any* of the m model entities, or form a NULL association. In the fully-explored tree, *without* pruning, and without heuristic termination after t non-NULL associations, there are $(m+1)^k$ nodes at the kth level, for a total tree size (assuming s scene

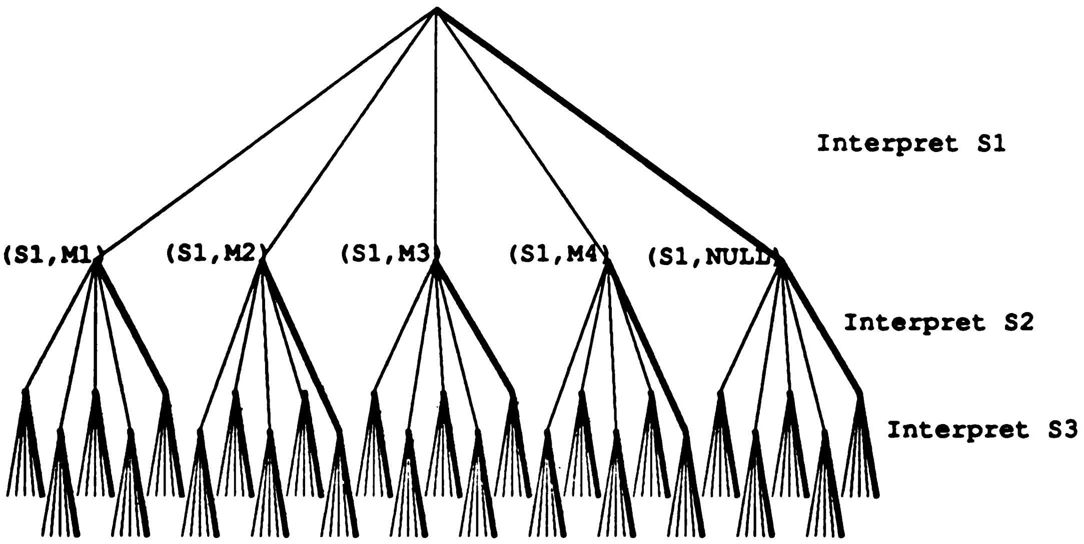

Figure 5.5: The full interpretation tree for $s = 3, m = 4$, allowing reuse of model entities, without heuristic search termination. Thick branches indicate NULL associations.

entities) of

$$\frac{(m+1)^{s+1}-1}{m}.$$

Figure 5.5 shows the interpretation tree constructed with $s = 3$, $m = 4$, allowing reuse of model entities, with no heuristic search termination. This tree has 156 nodes. The thick branches in the tree indicate NULL associations.

Suppose that we allow downward search to stop after t non-NULL associations have been obtained. In this case, we do not have a simple closed form for the number of nodes in the tree. We do have a recurrence formula, which can be tabulated for any desired values of m, s, and t:

$$\begin{aligned} N^*_{full}(S,M,T) &= 1 + mN^*_{full}(S-1,M,T-1) + N^*_{full}(S-1,M,T) \\ N^*_{full}(S,M,0) &= 1 \\ N^*_{full}(1,M,T) &= m+2. \end{aligned}$$

Figure 5.6 shows the interpretation tree generated for $s = 3, m = 4, t = 2$. Using the formula above, this tree has $N^*_{full}(3,4,2) = 76$ nodes.

In BONSAI, we design the search so that *model surfaces may not be re-used.* After an association involving model entity M has been created, any associations

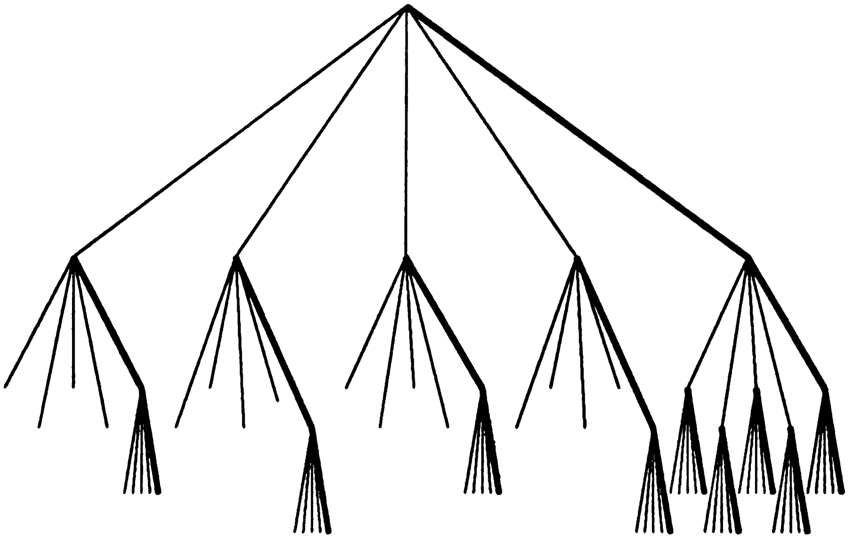

Figure 5.6: The full interpretation tree (with heuristic termination) for $s = 3, m = 4, t = 2$, allowing reuse of model entities.

created in a subtree of that hypothesis cannot involve M. We refer to this variant of the IT as a 'restricted interpretation tree.' Intuition should convince us that this modification can significantly reduce the size of the full IT, with or without heuristic search termination. The recurrence form for the restricted tree size, without heuristic search termination, is given by

$$\begin{aligned} N_{restricted}(S,M) &= 1 + mN_{restricted}(S-1,M-1) + N_{restricted}(S-1,M) \\ N_{restricted}(1,M) &= m+2 \\ N_{restricted}(S,0) &= s+1 \end{aligned}$$

Figure 5.7 shows the restricted tree without termination for $s = 3, m = 4$. The number of nodes is $N_{restricted}(3,4) = 100$.

When we add heuristic termination after t non-NULL associations, we have the following recurrence for the number of nodes in the IT:

$$\begin{aligned} N^*_{restricted}(S,M,T) &= 1 + mN^*_{restricted}(S-1,M-1,T-1) + N^*_{restricted}(S-1,M,T) \\ N^*_{restricted}(S,M,0) &= 1 \\ N^*_{restricted}(1,M,T) &= m+2 \\ N^*_{restricted}(S,0,T) &= s+1 \end{aligned}$$

Figure 5.8 shows the IT of this form for $s = 3, m = 4, t = 2$; it has 64 nodes.

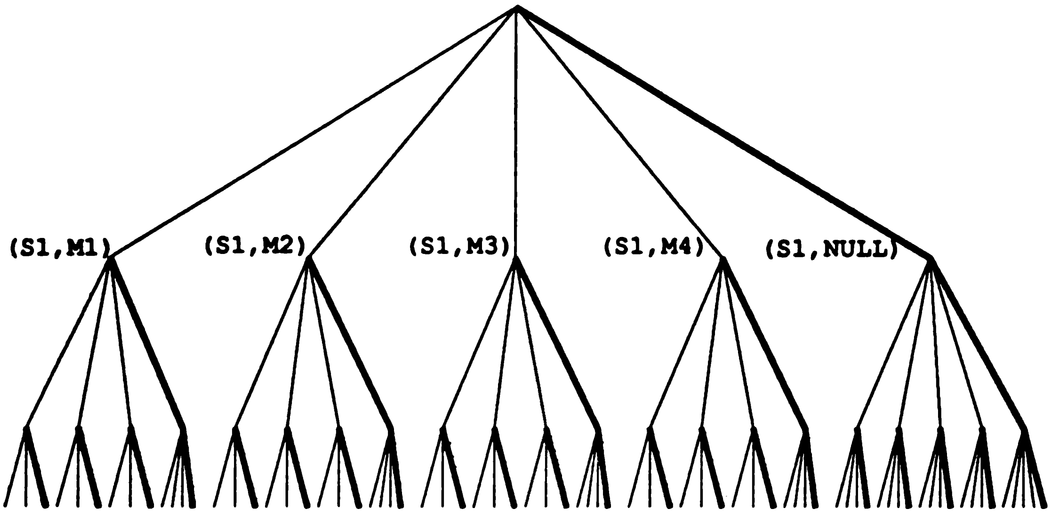

Figure 5.7: The restricted interpretation tree (without heuristic termination) for $s = 3, m = 4$.

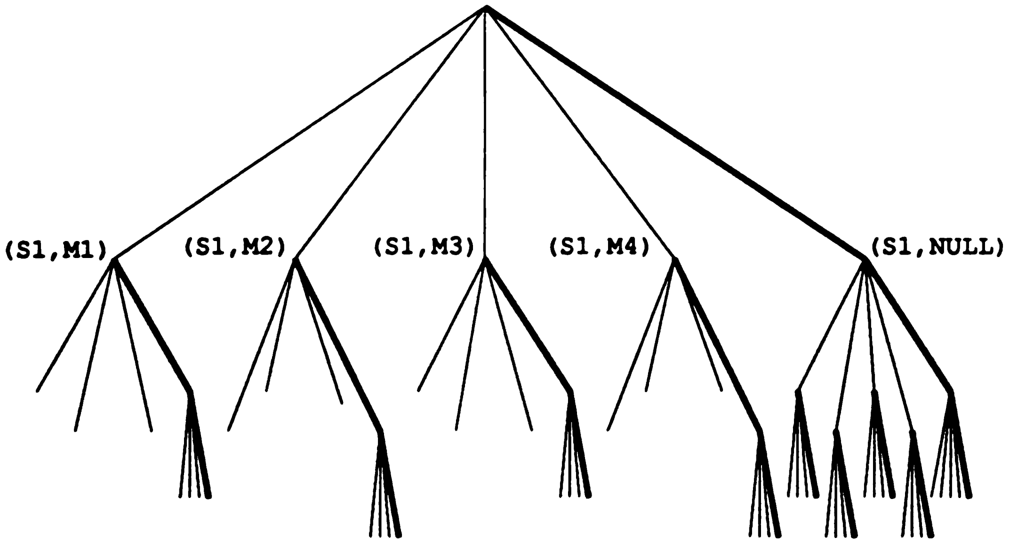

Figure 5.8: The restricted interpretation tree (with heuristic termination) for $s = 3, m = 4, t = 2$.

5.6.6 A Simulation Study

Our specification of Algorithm SEARCH and the probabilistic model of unary and binary consistency above allows us to simulate IT search under the model, and obtain rough bounds on the amount of search performed (the number of IT nodes visited) for a given combination of the probabilities p_i, the total number of scene entities s, the number of entities m in the model being searched, and the number of scene entities c ($c \leq s$) which actually come from the model under consideration. We limit our attention to search incorporating heuristic search termination, in which IT subtrees rooted at nodes with t non-NULL associations are not searched.

Figure 5.9 shows curves corresponding to the 5th and 95th percentiles of an empirical distribution of search simulation results for $p_1 = p_2 = 0.5, p_3 = p_4 = 0.99, m = 8$ with the horizontal axis being the value of s and c. Since $s = c$, we are implicitly simulating a search for an 8-sided isolated object under the assumption that segmentation is perfect, and has produced s segments, all of which match one of the 8 model faces. 100 simulations of the search were performed for each value of s. Each simulation proceeded as follows. The correct interpretation was specified by randomly choosing c of the s potential associations to be non-NULL and indexing them by an appropriate model entity. Then the search was initiated at the root node. When expanded, the unary and binary consistency were tested via Bernoulli trials. The total number of nodes expanded during search was tracked, and reported at the end of the simulation. Then the 100 numbers produced by the simulations were sorted, and the 5th and 95th in order were selected, and appear on the graph. In addition, there are isolated points on the graph corresponding to the amount of search performed for synthetic images of one of our database models, which happens to have eight faces. If the probabilistic model of consistency is correct, we would hope that the points on the plot would have a 90% chance of lying within the empirically-determined bounds. Our sample size is too low in this experiment to make broad statements. We simply note that three of the points were well outside the empirical bounds. Raising p_1 and p_2 would push the empirical bounds upward, and perhaps include the outliers. However, our previous remarks on the simplicity of the model definitely apply here.

Table 5.2 shows experimental results and empirical bounds for recognition of an object with 10 faces, several of which are curved. In these experiments, we set $p_1 = p_2 = 0.1, p_3 = p_4 = 0.99$, because experience has shown that both unary and binary

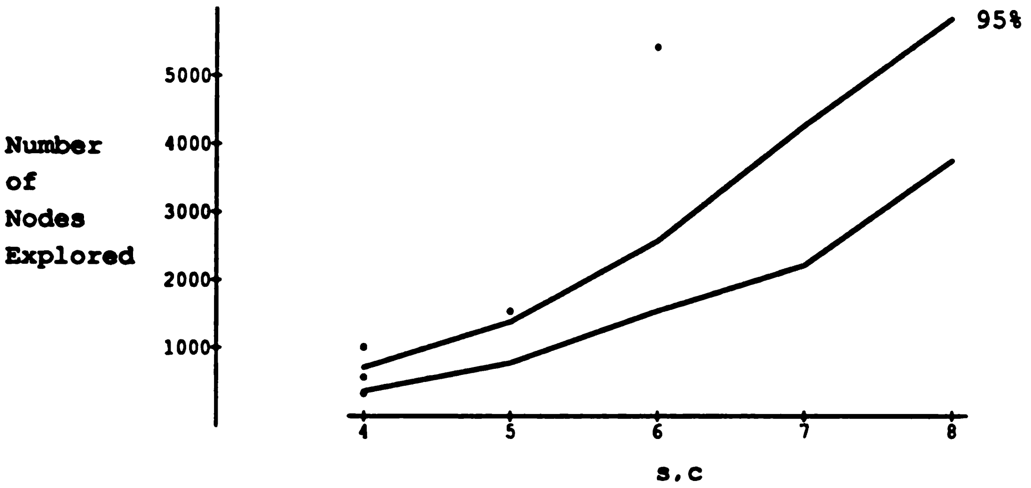

Figure 5.9: 5th and 95th percentile bounds on search (from simulation). Points are actual experiments with image data.

Table 5.2: Experimental results and empirical bounds for recognition of a curved object.

Experiment	s	m	c	result	lower bound	upper bound
1	7	10	5	287	487	918
2	14	10	3	2405	1057	2669
3	5	10	5	126	266	453
4	8	10	5	583	483	1082
5	5	10	5	187	276	483

constraints are more powerful with curved objects than with polyhedra. As above, the lower and upper bounds are obtained from the 5th and 95th percentile of a histogram of tree sizes determined from 100 simulations with the given parameters. The sizes from the actual experiments fall within the bounds in two of the five experiments, and lie below the lower bound in the other three.

Our conclusion regarding the probabilistic model of predicate behavior is that it can only be used as a rough guide for determining tree size. As mentioned earlier, the four-probability model is too simplistic to reflect the complexity of models, scenes, and noise sources. Extensions of the simulation-based approach to more realistic error models (with their parameters learned from object models) might offer a clearer look at search behavior under our set of constraints.

5.7 Summary

This chapter has described the BONSAI recognition module, which takes segmented scene descriptions, selects a candidate model subset from the object model database, and performs interpretation tree search, verifying hypotheses with adequately constrained pose estimates. Experiments with the system appear in Chapter 6. Our extension of IT search to work with curved surface primitives is one of the first reported in the literature. We have also developed the concept of *restricted* IT search, in which we allow model entities to be employed in hypothesis associations at most once. Restricted search trees are smaller and require less effort to search, and the underlying assumption of accurate segmentation for the scene can be relaxed through hypothesis extension.

Unlike some recent systems [27], BONSAI was not implemented with speed in mind; rather BONSAI serves as a powerful testbed for exploration of search-based scene interpretations involving geometric constraints. Once reliable segmentation procedures are developed for surface primitives such as general surfaces of revolution, or any curve or surface primitive described in terms of orientation and location parameters, BONSAI's constraints can be extended to accommodate those primitives. Paradoxically, BONSAI's worst performance is associated with polyhedral models, because the planar faces of polyhedra do not provide as many constraints on identity and pose as curved primitives. Redundant features (such as inter-patch angles) can lower the power of pruning constraints, causing many hypotheses (hundreds or thousands) to survive the search process. Since each hypothesis is expensive to verify, we would like as few as possible to survive the search.

One advantage of the hypothesis-verification procedure is large-grain parallelism. Since each verification is independent of all others, we could distribute the queue of surviving hypotheses to multiple computational platforms, have them perform the verification concurrently, and simply collect the hypothesis scores when they have finished. Alternatively, we could examine parallel implementations of the IT search itself. Kumar and Kanal [74] developed two parallel variants of the SSS* (state-space search) technique originally proposed by Stockman [107]. In one variant, multiple processors search the problem tree concurrently, 'competing' to find the optimal solution. In the other, the search space is partitioned into disjoint subsets, and a processor is dedicated to each subset. Application of parallel search methods to the IT search problem requires that a merit (or objective) function be available in order to compute the 'goodness' of a node in the tree. While the matching score developed in Section 5.5 might be usable as a merit function, it is only defined at terminal nodes in the search tree, and an extended definition would be required in order to prioritize the search at nonterminal nodes.

Recent efforts [45, 46] have focused on formal bounds on the effort expended during search. Tractable analysis requires a simple model of the error probabilities of pruning tests. We have conducted some preliminary simulations of an extended search model to obtain empirical bounds on recognition.

Chapter 6

Experiments

This chapter describes three series of experiments conducted with the BONSAI object recognition system. Twenty object models were used. Synthetic images of isolated objects allow us to estimate the accuracy of pose estimation. Real images of isolated objects are also examined, giving estimates of the error rate in recognition. Several images of occluded scenes were also processed; they illustrate BONSAI's ability to resolve multiple objects in a single scene.

3D object recognition is a popular research topic, and several systems have been described at length in the computer vision literature. However, many (if not most) of these systems are not thoroughly tested. Vision researchers have recently begun to address this shortcoming [52]. *At a minimum*, an object recognition system should be tested on 'many' images, to obtain statistically-reliable estimates of the various error rates (*e.g.* the incorrect-identification and false-reject error rates). It would be desirable for all images used in testing to be obtained from sensors rather than synthetically-generated. These goals obviously conflict with the expense often involved in taking images. As an attempt to address this problem, we have tested our system on over two hundred images, half being 'real' images (obtained from our 100X sensor), and half being synthetically generated. Thorough testing is *not* a substitute for good performance. The thoroughness of our tests means that we have a better grasp of the error rates, and not that those rates are lower than others. Comparison of BONSAI's accuracy with that of other systems must wait until those systems also undergo similar tests.

6.1 Pose Estimation Accuracy: Synthetic Imagery

Five images of each of our twenty models[1] were synthesized using the polygon scan-conversion approach described in Section 2.4.5. Representative views appear in Figure 2.20. These images were then segmented using the techniques outlined in Chapter 4. The resulting scene descriptions were presented to BONSAI, which was asked to estimate the pose of the object. No selection of candidate models was performed by the system; only the correct model was present in the database.

Tables 6.1 and 6.2 show the mean and standard deviations of errors in estimates of the rotation and translation for nonsymmetric and rotationally-symmetric objects, respectively. Note that for nonsymmetric objects, the pose rotation can be expressed uniquely as a 3D vector $\mathbf{v}$ through the object, an angle of rotation θ about $\mathbf{v}$, and a translation $\boldsymbol{\Delta} = (\Delta x, \Delta y, \Delta z)$. If we denote the corresponding estimates as $\hat{\mathbf{v}}, \hat{\theta}$, and $\hat{\boldsymbol{\Delta}}$, respectively, the three columns of numbers in Table 6.1 are the mean and standard deviation of $\cos^{-1}(\mathbf{v} \cdot \hat{\mathbf{v}})$ (in degrees), $\mid \theta - \hat{\theta} \mid$ (in degrees), and $\sqrt{\|\boldsymbol{\Delta} - \hat{\boldsymbol{\Delta}}\|}$ (in inches), for five views of the model. For rotationally-symmetric objects, one of the three components of the rigid rotation is undefined, and we report the mean and standard deviation of the angle (in degrees) between the known axis of symmetry and the estimated axis of symmetry. For the two spherical models (BALL and BALL2), only the mean and standard deviation of the translation error is reported, as the orientation is not uniquely determined.

For most of our objects, the accuracy of pose estimates is very good (within 1° for angles and 0.1 inch for translations). In our experiments, those models producing large average pose errors actually were misidentified in some of the views. Therefore, the pose estimate could not be expected (in general) to be correct anyway. If higher accuracy were desired, these estimates could serve as input to an iterative pose-improvement procedure which would attempt small adjustments in the angles and translations to optimize a matching criterion. Alternatively, another sensor (perhaps a tactile sensor) could be used to locate the object to the desired precision.

[1] See Section 2.5 for descriptions of the object models.

Table 6.1: Nonsymmetric objects: mean and standard deviation of errors in axis orientation, axis rotation, and translation.

Object	Axis orientation (degrees)		Rotation about axis (degrees)		Translation (inches)	
	Mean	Std. dev.	Mean	Std. dev.	Mean	Std. dev.
BALL	-	-	-	-	0.020	0.001
BALL2	-	-	-	-	0.018	0.002
BIGWYE	5.614	6.098	0.762	1.159	0.488	0.505
BLOCK1	0.162	0.165	0.087	0.128	0.014	0.007
BLOCK2	0.332	0.486	0.299	0.300	0.022	0.017
BLOCK4	0.570	0.871	0.644	1.212	0.029	0.041
BOX2INCH	1.479	0.842	1.602	0.934	0.086	0.007
COLUMN1	0.337	0.353	0.584	0.508	0.026	0.025
COLUMN2	0.334	0.232	0.333	0.263	0.027	0.007
CURVBLOCK	0.216	0.221	0.279	0.303	0.158	0.028
GRNBLK3	0.668	0.0294	0.231	0.251	0.014	0.007
HUMP	0.237	0.492	0.822	0.772	0.172	0.008

Table 6.2: Symmetric objects: mean and standard deviation of errors in axis of symmetry and translation.

Object	Axis of symmetry orientation (degrees)		Translation (inches)	
	Mean	Std. dev.	Mean	Std. dev.
ADAPTER	0.537	0.588	0.022	0.009
AGPART2	2.892	4.158	0.059	0.031
CAP	0.081	0.045	0.417	0.611
HALFSPH	0.059	0.050	0.008	0.002
HARRISCUP	0.103	0.031	0.028	0.010
PISTON	0.086	0.038	0.039	0.024
PROPANE	0.078	0.040	0.066	0.0534
TAPEROLL	0.079	0.064	0.133	0.100

6.2 Recognition Accuracy: Real Imagery

Five images of each of our twenty objects were obtained from the 100X scanner and segmented using the techniques in Chapter 4. During the image acquisition process, we took care to position the objects so that enough faces were visible to provide us with an adequately constrained pose estimate. When scanning polyhedra, for example, we tried to ensure that at least three non-coplanar faces were visible. This restriction on image acquisition could be relaxed by incorporating knowledge about the bounds (in the image plane) on the object surfaces. This is a topic for future research, however; at present, we employ only the geometric parameters in the pose estimation procedure.

The resulting scene descriptions were input to BONSAI, which selected a subset of models to search and produced a ranked list of recognition hypotheses. The confusion matrix in Table 6.3 displays the correct and incorrect identities of the *top-ranked* hypotheses for each experiment. A nonzero number m_{rc} in the matrix at row r and column c means that m_{rc} number of views of model r were identified by BONSAI as model c. Correct interpretations appear on the diagonal of the matrix, while off-diagonal numbers indicate misinterpretations. In addition, there is an extra column in the matrix, corresponding to the reject option, where no hypotheses were generated. Summarizing the results, we have an overall recognition accuracy of 88%, a false reject rate of 2%, and an incorrect recognition rate of 10%. These error rates are comparable to those obtained by Lu *et al.* [79] from their CAD-based vision system (using intensity data). Hoffman *et al.* [55] obtained perfect error rates on a twelve-object database, but used only synthetic data. Instead of accepting *only* the top-ranked recognition hypothesis, we could select the top k hypotheses. In 92% of our test views, the correct hypothesis was top-ranked *or* second-ranked. In 95% of our test views, the correct hypothesis was among the top three hypotheses ranked by the matching score.

Most of the errors made by our system were *plausible* interpretations of the scene. For example, many of the polyhedral or partially polyhedral objects in our database exhibit views which are qualitatively similar, and we were not surprised to see our system have difficulty recognizing the correct model in such ambiguous cases. Many of our polyhedral models have 90° dihedral edges and corners; in some views containing only three planes (*i.e.* with only one visible corner), the system produces many candidate hypotheses, which match that corner equally well, and will not be rejected

Table 6.3: Confusion matrix for recognition accuracy experiments.

		Classification																				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	REJECT
	1	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	1	0	0	0	0
	9	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0
	10	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0
Object	11	0	0	0	0	0	0	0	0	0	0	4	1	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
	13	0	1	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0
	14	0	0	0	0	0	0	1	0	0	0	0	0	0	4	0	0	0	0	0	0	0
	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
	16	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	1
	17	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0
	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	3	0	0	1
	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	3	0	0
	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0

Row and Column Indices:
1: ADAPTER, 2: AGPART2, 3: BALL, 4: BALL2, 5: BIGWYE, 6: BLOCK1
7: BLOCK2, 8: BLOCK4, 9: BOX2INCH, 10: CAP, 11: COLUMN1
12: COLUMN2, 13: CURVBLOCK, 14: GRNBLK3, 15: HALFSPH
16: HARRISCUP, 17: HUMP, 18: PISTON, 19: PROPANE, 20: TAPEROLL.

by the geometric constraints. Alternative area-based or edge-based matching scores could be used in the place of the technique given in Section 5.5, which compares the range data and segmentations but not surfaces or edges.

Figure 6.1 illustrates one of the misinterpretations for the TAPEROLL object. Figures 6.1(a), (b), and (c) show the input range image, pseudo-intensity image, and segmentation image. Figures 6.1(d), (e), and (f) show the range, pseudo-intensity, and segmentation images for the highest-ranked hypothesis, involving the PISTON object (which has a similar radius to the outside surface of the TAPEROLL object). The score for this hypothesis was 0.52. Figures 6.1(g), (h), and (i) illustrate one of the correct hypotheses generated by BONSAI; its score was 0.28.

6.3 Cluttered Scene Processing

Ten images, each containing two model objects, were obtained from the 100X sensor, processed, and input to BONSAI, which produced a ranked list of hypotheses regarding the image content. We saved the first hypothesis and removed its associated

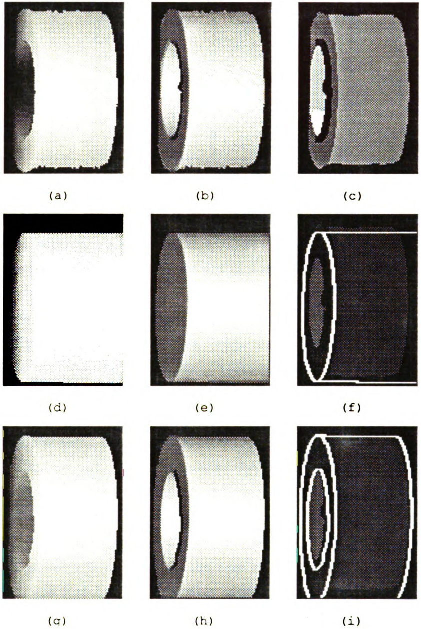

Figure 6.1: A misinterpretation of the TAPEROLL object.

Table 6.4: Recognition results for ten cluttered scenes.

Image name	Figure number	Number of correct interpretations	Number of subjectively good pose estimates
ADAPTER/CURVBLOCK	6.2	2	2
BALL2/PROPANE	6.3	2	2
BIGWYE/TAPEROLL	6.4	2	2
BLOCK2/HARRISCUP	6.5	2	2
BLOCK4/HALFSPH	6.6	2	2
CAP/BOX2INCH	6.7	0	0
COLUMN1/BLOCK1	6.8	0	0
COLUMN2/BALL	6.9	1	1
GRNBLK3/PISTON	6.10	2	1
HUMP/AGPART2	6.11	2	2

scene entities from the system's scene description. BONSAI then produced another list of hypotheses, and we saved the highest-ranked of these. In cases where we do not know the number of objects present in the scene, a different strategy, such as selecting maximally-disjoint interpretations from the output of a *single* run of BONSAI, can be used. Since we knew *a priori* the number of objects in each test image, we chose to use this information during interpretation.

Table 6.4 summarizes the results of these ten recognition experiments. Figures 6.2 through 6.11 show the input range images, pseudo-intensity images, segmentations, and top and second-ranked hypotheses for the experiments.

In the ADAPTER/CURVBLOCK image, the original segmentation contained twelve patches, and the pruning constraints proved too weak to effectively trim the interpretation tree. Consequently, several thousand hypotheses of identity and position for the CURVBLOCK object were produced. In order to obtain results in a reasonable amount of time, we removed six of the planar patches from the original segmentation and ran BONSAI on the remainder. In the CAP/BOX2INCH image, neither hypothesis of identity is correct. In the case of the cube, a corner of the CURVBLOCK object was matched to the visible corner of the cube and produced a higher hypothesis score (0.213), although the correct hypothesis of identity had a score which was only slightly lower (0.184). More *a priori* knowledge about the sensing environment could allow us to reject the CURVBLOCK hypothesis; for example, we could specify that the images contain the entire model. In that case, we would reject CURVBLOCK as a candidate model since the field of view is smaller than the hypothesized model. The CAP object in the scene was identified as the

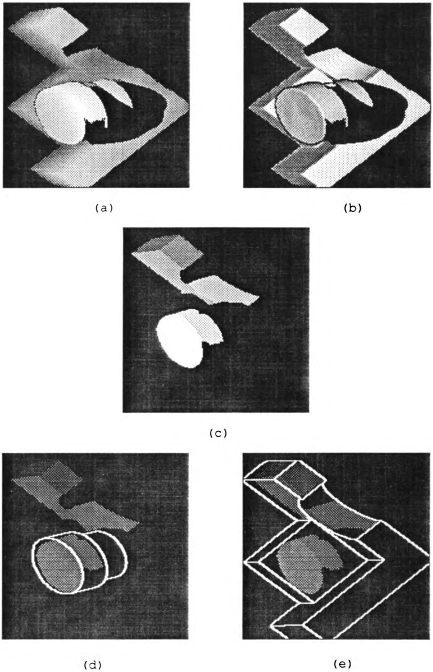

Figure 6.2: Recognition results for ADAPTER/CURVBLOCK scene. (a): range image. (b): pseudo-intensity image. (c): segmentation (modified; see text). (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

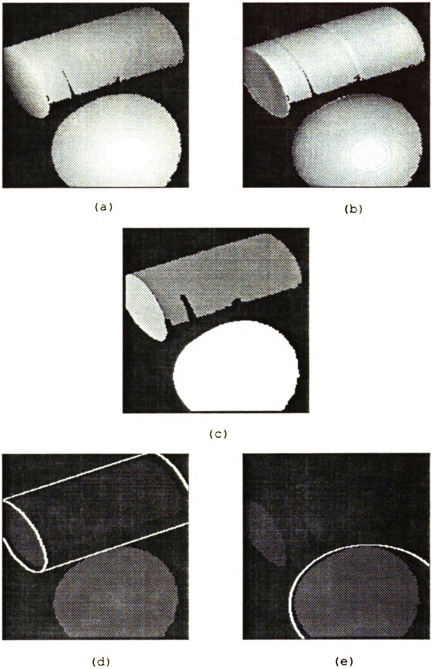

Figure 6.3: Recognition results for BALL2/PROPANE scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

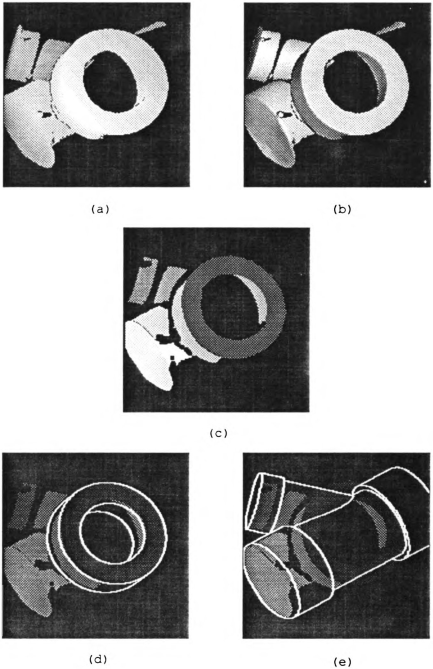

Figure 6.4: Recognition results for BIGWYE/TAPEROLL scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

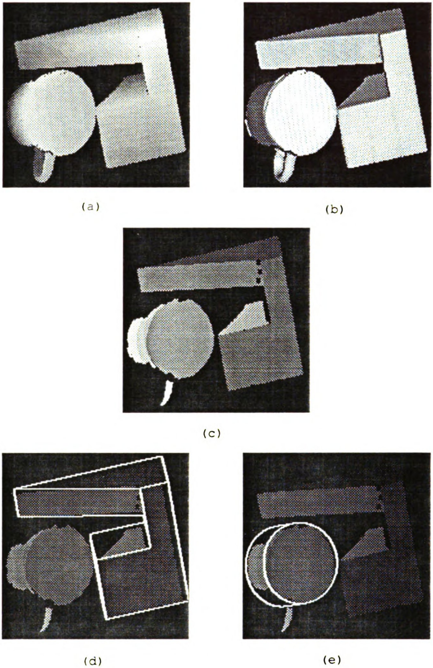

Figure 6.5: Recognition results for BLOCK2/HARRISCUP scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

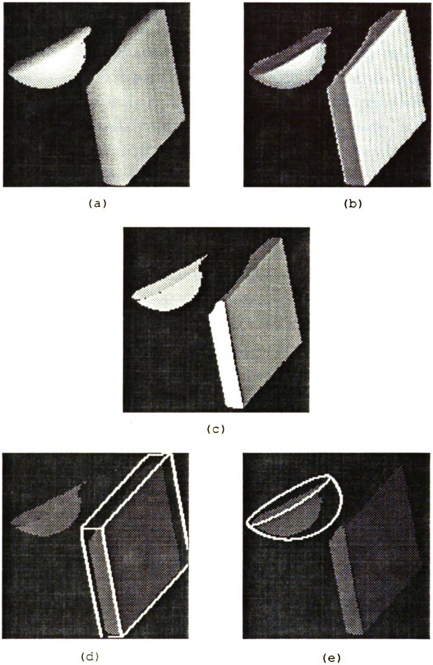

Figure 6.6: Recognition results for BLOCK4/HALFSPH scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

ADAPTER model; the ADAPTER hypothesis score was 0.122, while the correct hypothesis' score was 0.106. Similar radii and identical angle relations between the planar and cylindrical patches caused both models to be considered.

The COLUMN1/BLOCK1 image in Figure 6.8 produced hypotheses which, while incorrect, are plausible and highlight additional ways to extend BONSAI's use of constraints. The two highest-ranked hypotheses match the COLUMN1 objects against the COLUMN2 model, and the BLOCK1 hypothesis has a reflected orientation estimate. The COLUMN2 interpretation is plausible since such an image could be produced by occlusion of part of COLUMN2 by BLOCK1. The correct hypothesis involving COLUMN1 had a score of 0.082, versus a score of 0.085 for COLUMN2. One possible way to prune this sort of interpretation would involve *configuration understanding*, in which the system, using interference tests and an *a priori* rule which states that objects cannot interpenetrate, would reject the COLUMN2 hypothesis because the scene content indicates that the other scene patches cannot be in their observed positions under that hypothesis. The three largest patches from the image of BLOCK1 were incorrectly assigned to the BLOCK1 model. Since the patches involved in this interpretation are planar and mutually orthogonal, this interpretation is plausible, and the absence of the slanted patch in the input image eliminated a constraint that otherwise would have pruned the interpretation. The correct interpretation had a score of 0.135, as opposed to the incorrect interpretation's score of 0.147. The triangular-shaped patch in the input image has been matched to a rectangular model surface. Assuming that 2D shape could be reliably extracted from the segmentation images, an additional (unary) predicate (namely, a patch-shape predicate) could increase the power of BONSAI in these situations.

In the COLUMN2/BALL image, the ball was correctly recognized, but the COLUMN2 object was misidentified as an instance of CURVBLOCK, based on an incorrect assignment between the three planar faces forming the upper right corner of the object. The correct hypothesis of identity had a score of 0.030 as opposed to the incorrect hypothesis' score of 0.053. The addition of the cylindrical patch from COLUMN2 to a correct hypothesis of identity would provide an additional constraint, which could raise the correct hypothesis' score. In this image (as in the previous image), the right-angle corner feature involved in the interpretation is not particularly salient (discriminatory) for many of the models. Improvements of the search procedure would concentrate on more salient features, and use the nonsalient artifacts only

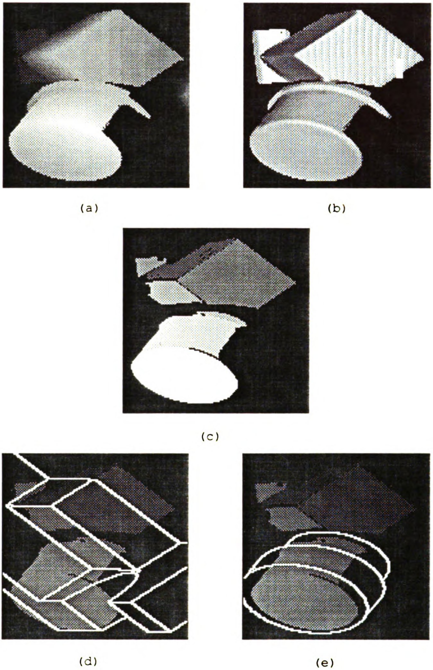

Figure 6.7: Recognition results for CAP/BOX2INCH scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

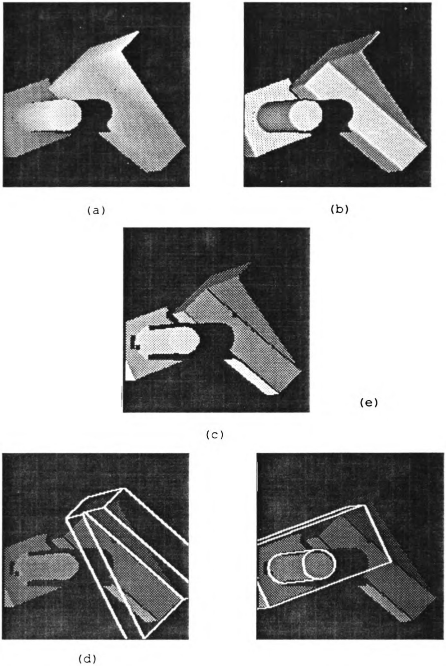

Figure 6.8: Recognition results for COLUMN1/BLOCK1 scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

during verification.

In the GRNBLK3/PISTON image, the PISTON object was identified correctly, but the GRNBLK3 hypothesis, while correct in terms of object identity, also exhibits the problem with a nonsalient corner feature matching against the wrong planar patches. Only one patch for this object has discriminatory angle relations with neighboring patches, and its relatively small area caused it to be left out of the initial IT search.

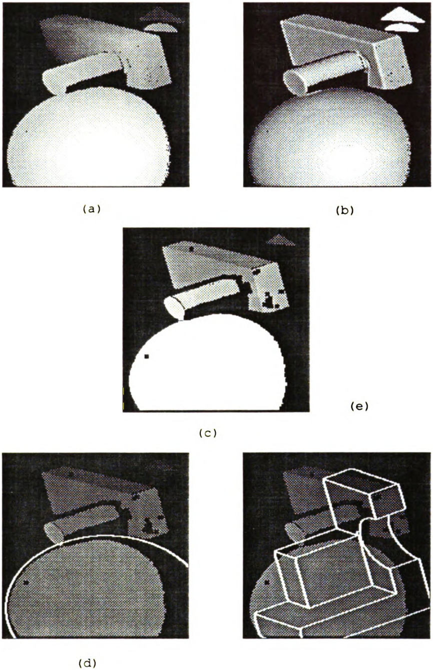

Figure 6.9: Recognition results for COLUMN2/BALL scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

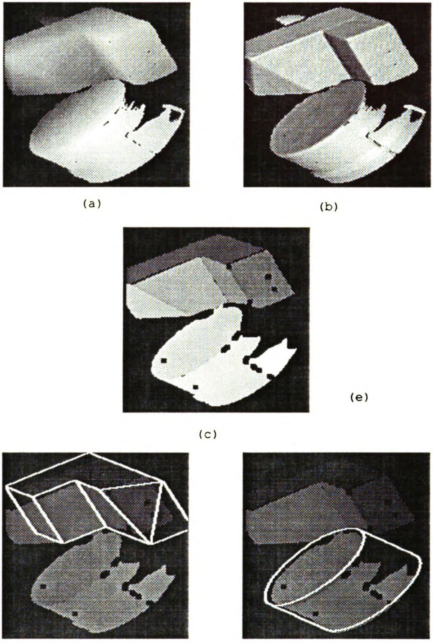

Figure 6.10: Recognition results for GRNBLK3/PISTON scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

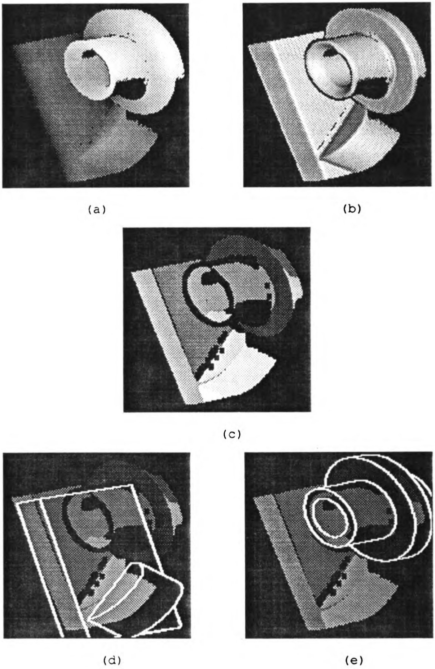

Figure 6.11: Recognition results for HUMP/AGPART2 scene. (a): range image. (b): pseudo-intensity image. (c): segmentation. (d): Top-ranked hypothesis. (e): Second-ranked hypothesis.

6.4 Discussion

Experiments with synthetic and real data indicate that BONSAI can produce correct scene interpretations in situations where views are not underconstrained (in terms of pose) and models have a significant number of salient features. Models or scenes with structural similarities or nonsalient features can occasionally yield incorrect interpretations, but those interpretations are usually plausible nonetheless. As a testbed for constraint-based IT search, BONSAI is a success, in our opinion. However, BONSAI is not particularly fast. Scenes can take from seconds to hours (even days!) to interpret, depending on the number of hypotheses generated, the scene complexity, the saliency of unary and binary relations, and the size of the model database. Since curved surfaces provide more constraints on interpretation, BONSAI (unlike most other systems we are aware of) typically resolves the identity of nonpolyhedral objects more quickly than polyhedra. This suggests that BONSAI could be extended through the addition of constraints designed specifically for polyhedral models, which would improve the system's overall speed.

Our present set of experiments yields a recognition error rate for isolated objects which is comparable with that of other systems being described in the literature, pose estimates of good quality, and marginal interpretation quality for cluttered scenes, although additional constraints (specifically, the computation of feature saliency) can be employed to improve the results.

Chapter 7

Conclusions and Recommendations for Future Research

This thesis has explored the problem of 3D object recognition using CAD models as a source of part geometry. Given a depth map of a scene containing (in general) several parts, the BONSAI system derives a symbolic description of the image and forms hypotheses of part identity and pose (location and orientation) using heuristic search constrained by unary and binary geometric predicates, followed by hypothesis verification using synthetic imagery. In experiments on over two hundred images (both real and synthetic) of twenty object models, the system exhibited 88% accuracy in identification of isolated objects, and produced reasonably accurate pose estimates (translations within a few tenths of an inch and rotations within a few degrees). Incorrect scene interpretations are usually plausible interpretations of the scene. In isolated-object recognition experiments, 95% of the correct hypotheses were among the three highest-ranked hypotheses for each image.

The primary contributions of this research are a methodology for automatically converting solid models from CAD systems to relational graphs which are used by the object recognition module, and the recognition system itself, which advances the interpretation tree paradigm for 3D recognition into the domain of curved surfaces. BONSAI's major shortcomings are:

- a brittle, data-driven segmentation procedure;
- a time-consuming verification process;
- weakness of search predicates for certain object types; and
- the inherent exponential time complexity of IT search.

Clearly, the domain of automated manufacturing constrains the types of images we would expect to obtain from a range sensor. One area we will address in the future is the integration of knowledge about part geometry into the segmentation procedure. For example, if the parts in the current model database contained only a few cylindrical patches with specified radii, we could tailor a model-driven segmentation system with specialized 'cylinder-finding' modules appropriate to the database. Customized segmentation procedures offer increased robustness to noise in many cases, since the dimension of the parameter spaces involved in surface fitting will often be lower for curved surfaces. In addition, *the successful extraction of a certain surface imposes constraints on interpretation.* If our model database contains only one cylindrical patch with a radius of 1 inch, the successful extraction of that patch *immediately* suggests that the corresponding model is present in the scene. In this way, we can constrain interpretation tree search *before the fact* in addition to the pruning done within the search module.

A large number of hypotheses can be generated from some scenes. At present, we verify hypotheses by generating a synthetic range and segmentation image from the CAD model and computing an area-based matching score. Other verification approaches are possible, however. Instead of matching on patch areas (which is time-consuming), we could match on region boundaries (edges) by accumulating the distances between observed edge pixels for each pair of patches and their predicted locations (obtainable from the pose estimate and coordinates of edge points obtained from the CAD model).

In Chapter 5, we stated that verifications can be performed in parallel, since each hypothesis is independent of the others. We could also save time by reducing the resolution of the synthetic imagery generated during verification (this might be termed *coarse verification*). Only hypotheses with high scores from the coarse verification module would be verified at the same resolution as the input image. Alternative approaches to verification can be explored as well. Our difficulties in extracting crease edges from range data precluded exploration of edge-based verification; however, if a model-driven segmentation procedure were developed, we might be able to extract and accurately characterize creases and jump edges, and use them *instead* of patches as verification entities.

Our object database contained several polyhedral (or mostly polyhedral) objects with 90 degree angles between adjacent faces. BONSAI typically produces a large number of hypotheses (hundreds or thousands) for such objects. The reasons for this behavior are twofold. First, planar surfaces inherently provide fewer opportunities for search tree pruning than curved surfaces. Planes do not possess intrinsic parameters other than their planarity, so only type, area, visibility, and orientation pruning can be performed. BONSAI depends heavily on orientation relations to prune the search tree, and for the boxlike objects with similar orientation relations between faces, the orientation predicate is much less powerful. To address this problem, we might wish to relax our heuristic termination requirement. Instead of terminating a search once a consistent hypothesis with three associations has been found, we can let search proceed one or more levels below the termination point, in the hope that more pruning would take place, producing a smaller number of net hypotheses. For the polyhedral cases, this enlargement of the search tree might significantly reduce the number of surviving hypotheses.

Both theoretical studies [45] and practical experience show that recognition via search has exponential time complexity in cluttered scenes. Instead of making modifications to the IT search procedure, we might wish to abandon a search-based approach entirely. *Geometric hashing* [75, 53, 27] is a relatively new approach to recognition, in which a hash table indexed by invariants computed from subsets of model entities is created off-line. The hash table entries consist of the model identity and a pose estimate. Recognition is performed by computing similar invariants from the input scene, and extracting the model identity and pose from the table. Although previous work has focused primarily on 2D models and scenes, extension of the work to 3D might produce systems which can recognize objects in polynomial time. The time and space consumed during construction of the hash table might be a barrier to practical implementation of a hashing-based recognition system. It is important to note that IT search is not the only way to recognize 3D objects in 3D scenes.

Appendix

Appendix A

Program Input and Output

A.1 Model-builder

Figure A.1 shows a portion of the IGES description of the GRNBLK3 model. The comments in square brackets are not part of the file; they explain the meaning of some of the entities. Selected lines from the LISP code generated by the geometric inference engine appear in Figure A.2. In both figures, deleted sections are marked with ellipsis (...).

A.2 Segmentation Output

Figure A.2 shows the LISP code generated from the segmentation and segment classification modules of the vision system. The input image appears in Figure 4.2. Figure A.3 gives the LISP code generated by the segmentation and classification module using the range image of Figure 4.8.

```
[header]
GEOMETRY CREATED BY SDRC GEOMOD 4.0                                    S      1
,,7HGRNBLK3,7HGRNBLK3, 36HSDRC GEOMOD;V4.0 ,SUN   ,TYPE-IGES30,,32,8,24,G      1
8,56,7HGRNBLK3,1.0,1,2HIN,,,13H891029.140737,1.0E-05,4.4E+00,,;        G      2
[directory entry for line 1]
    110      1      1      1      0      0      0      000000000D      1
    110      0      7      2      0                           0D      2
...
[directory entry for plane 49]
    108     49      1      1      0      0      0      000000001D     49
    108      0      7      3      1                           0D     50
...
[directory entry for plane 145]
    108    153      1      1      0      0      0      000000001D    145
    108      0      7      3      1                           0D    146
[geometric parameters for line 1]
110, 0.4370000E+01, 0.1570000E+01, 0.2870000E+01,                      1P      1
 0.3100000E+01, 0.1570000E+01, 0.2923353E-17, 0, 0;                    1P      2
...
[geometric parameters for plane 49]
108, 0.0000000E+00, 0.1000000E+01, 0.0000000E+00, 0.1570000E+01,      49P     49
   47, 0.0000000E+00, 0.0000000E+00, 0.0000000E+00,                   49P     50
 0.0000000E+00,0,0;                                                   49P     51
...
[geometric parameters for plane 145]
108, 0.7351827E+00, 0.5947018E+00,-0.3253247E+00, 0.3212748E+01,     145P    153
  143, 0.0000000E+00, 0.0000000E+00, 0.0000000E+00,                  145P    154
 0.0000000E+00,0,0;                                                  145P    155
[end-of-file record]
S      1G      2D     146P     155                                     T      1
```

Figure A.1: A portion of the IGES file for the polyhedron

```
(setq grnblk3-line1 (make-line :start '(4.370000 1.570000 2.870000)
  :end '(3.100000 1.570000 0.000000) :length 3.138439 :transformation NIL
  :visible t :parent 'grnblk3))
(setq grnblk3-lines (cons 'grnblk3-line1 grnblk3-lines))
...
(setq grnblk3-plane49 (make-plane
  :coefficients '(0.000000 1.000000 0.000000 1.570000)
  :bounding-curve 'grnblk3-composite-curve47
  :canonical-transformation (make-array '(4 4) :initial-contents
  '((0.000000 -0.000000 -1.000000 0.000000)
    (-1.000000 -0.000000 0.000000 0.000000)
    (-0.000000 1.000000 -0.000000 0.000000)
    (0.000000 0.000000 0.000000 1.000000)
    ))
  :canonical-bounding-box '((-4.950000 0.150000)(-6.700001 -1.600000))
  :transformation NIL :visible t :parent 'grnblk3))
(setq grnblk3-planes (cons 'grnblk3-plane49 grnblk3-planes))
...
(setq grnblk3-plane145 (make-plane
  :coefficients '(0.735183 0.594702 -0.325325 3.212748)
  :bounding-curve 'grnblk3-composite-curve143
  :canonical-transformation (make-array '(4 4) :initial-contents
  '((-0.252932 -0.204601 -0.945602 0.000000)
    (-0.628913 0.777476 0.000000 0.000000)
    (0.735183 0.594702 -0.325325 0.000000)
    (0.000000 0.000000 0.000000 1.000000)
    ))
  :canonical-bounding-box '((-6.857789 2.484998)(-4.710185 4.632603))
  :transformation NIL :visible t :parent 'grnblk3))
...
(setq grnblk3-planes (cons 'grnblk3-plane145 grnblk3-planes))
(setq grnblk3 (make-object
  :circular-arcs grnblk3-circular-arcs
  :composite-curves grnblk3-composite-curves
  :planes grnblk3-planes
  :lines grnblk3-lines
  :parametric-spline-curves grnblk3-parametric-spline-curves
  :ruled-surfaces grnblk3-ruled-surfaces
  :surfaces-of-revolution grnblk3-surfaces-of-revolution
  :tabulated-cylinders grnblk3-tabulated-cylinders
  :rational-b-spline-curves grnblk3-rational-b-spline-curves
  :rational-b-spline-surfaces grnblk3-rational-b-spline-surfaces
  :curves grnblk3-curves
  :surfaces grnblk3-surfaces))
...
(angle 'grnblk3-plane49 'grnblk3-line1 1.570796)
(adjacent-to 'grnblk3-plane49 'grnblk3-plane61)
(edge-info 'grnblk3-plane49 'grnblk3-plane61
 '(1.650000 1.570000 2.870000) '(1.650000 1.570000 0.000000)
 '((1.650000 1.650000)(1.570000 1.570000)(0.000000 2.870000)))
(setf (nth 0 (generic-area grnblk3-plane49)) 5.646171)
(setf (nth 1 (generic-area grnblk3-plane49)) 5.678042)
(setf (nth 2 (generic-area grnblk3-plane49)) 5.426592)
...
```

Figure A.2: S-expressions generated automatically from the polyhedral model

```
(setq *scene* nil)
(setq scene-plane1 (make-plane :coefficients '(0.174419 0.120727 0.977243 -1.299193)))
(setq *scene* (append *scene* '(scene-plane1)))
(setq scene-plane2 (make-plane :coefficients '(-0.612083 0.426335 0.666028 1.535971)))
(setq *scene* (append *scene* '(scene-plane2)))
(setq scene-plane3 (make-plane :coefficients '(0.169948 0.122183 0.977849 0.291045)))
(setq *scene* (append *scene* '(scene-plane3)))
(setq scene-plane4 (make-plane :coefficients '(-0.775188 -0.591141 0.222793 -1.482211)))
(setq *scene* (append *scene* '(scene-plane4)))
(setq scene-plane5 (make-plane :coefficients '(-0.934494 0.207226 0.289445 -1.343550)))
(setq *scene* (append *scene* '(scene-plane5)))
(setq scene-plane6 (make-plane :coefficients '(-0.930482 0.208735 0.301054 1.346091)))
(setq *scene* (append *scene* '(scene-plane6)))
(setq scene-plane7 (make-plane :coefficients '(0.307099 -0.069145 0.949162 -0.376711)))
(setq *scene* (append *scene* '(scene-plane7)))
(setq scene-surface-of-revolution8 (make-surface-of-revolution
  :surface-of-revolution-type 'cylinder
  :radius 0.660118 :direction '(-0.920777 0.223857 0.319466)
  :possible-points-on-axis '(0.355934 0.533573 -1.862916)))
(setq *scene* (append *scene* '(scene-surface-of-revolution9)))
(setf (plane-area scene-plane1) 4.759323)
(setf (plane-area scene-plane2) 2.300559)
(setf (plane-area scene-plane3) 3.819569)
(setf (plane-area scene-plane4) 4.471331)
(setf (plane-area scene-plane5) 2.630743)
(setf (plane-area scene-plane6) 0.935534)
(setf (plane-area scene-plane7) 0.038796)
(setf (surface-of-revolution-area scene-surface-of-revolution8) 2.483092)
(img-adjacent 'scene-plane1 'scene-plane2)
(img-adjacent 'scene-plane2 'scene-plane4)
(img-adjacent 'scene-plane1 'scene-plane4)
(img-adjacent 'scene-plane3 'scene-plane4)
(img-adjacent 'scene-plane5 'scene-plane7)
(img-adjacent 'scene-plane5 'scene-surface-of-revolution8)
(scene-edge-info 'scene-plane1 'scene-plane2
  '((-1.431015 3.907000 -1.583000)(-0.287024 6.345000 -2.068000) 0.000266)
  '((-1.431015 -0.287024)(3.907000 6.345000)(-2.068000 -1.583000)))
(scene-edge-info 'scene-plane1 'scene-plane4
  '((-0.703021 2.940000 -1.568000)(-1.431015 3.907000 -1.583000) 0.000679)
  '((-1.431015 -0.703021)(2.940000 3.907000)(-1.583000 -1.563000)))
(scene-edge-info 'scene-plane2 'scene-plane4
  '((-2.185008 4.499000 -2.598000)(-1.457014 3.907000 -1.584000) 0.001136)
  '((-2.263008 -1.457014)(3.907000 4.599000)(-2.735000 -1.584000)))
(scene-edge-info 'scene-plane3 'scene-plane4
  '((0.336971 2.194000 -0.036000)(-0.183025 2.856000 -0.032000) 0.000457)
  '((-0.183025 0.336971)(2.194000 2.856000)(-0.036000 -0.029000)))
(scene-edge-info 'scene-plane5 'scene-plane7
  '((0.986965 -0.851000 -0.866000)(1.532961 1.499000 -0.821000) 0.000404)
  '((0.986965 1.532961)(-0.851000 1.499000)(-0.880000 -0.821000)))
(scene-edge-info 'scene-plane7 'scene-surface-of-revolution8
  '((1.038965 -0.891000 -0.821000)(1.558961 1.499000 -0.812000) 0.000176)
  '((1.038965 1.558961)(-0.891000 1.499000)(-0.821000 -0.810000)))
(setq *scene-rows* 182)
(setq *scene-columns* 208)
(setq *scene-x-resolution* 0.026000)
(setq *scene-y-resolution* 0.046093)
; yres min 0.039000, max 0.116000, mean 0.046093, var 0.000009, n 17728
(setq *scene-point* '((71 79) (-0.521022 3.540000)))
(setq *scene-label-file* "/home/surya/flynn/cad/imgs/misc/jumble12/jumble12.label4.Z")
(setq *scene-image-file* "/home/surya/flynn/cad/imgs/misc/jumble12/jumble12.hips.Z")
```

```
(setq *scene* nil)
(setq scene-plane1 (make-plane :coefficients '(-0.751939 0.477385 0.454634 1.843183)))
(setq *scene* (append *scene* '(scene-plane1)))
(setq scene-plane2 (make-plane :coefficients '(0.394840 -0.244066 0.885738 -3.421716)))
(setq *scene* (append *scene* '(scene-plane2)))
(setq scene-plane3 (make-plane :coefficients '(0.386544 -0.240616 0.890330 -3.419152)))
(setq *scene* (append *scene* '(scene-plane3)))
(setq scene-plane4 (make-plane :coefficients '(-0.754584 0.473868 0.453930 0.366817)))
(setq *scene* (append *scene* '(scene-plane4)))
(setq scene-plane5 (make-plane :coefficients '(0.142912 -0.570143 0.809020 -2.976815)))
(setq *scene* (append *scene* '(scene-plane5)))
(setq scene-plane6 (make-plane :coefficients '(0.220402 0.864034 0.452625 1.839125)))
(setq *scene* (append *scene* '(scene-plane6)))
(setq scene-surface-of-revolution7 (make-surface-of-revolution
 :surface-of-revolution-type 'cylinder :radius 1.046759
 :direction '(0.230747 0.862543 0.450307)
 :possible-points-on-axis '(-1.172338 3.024273 -0.037951)))
(setq *scene* (append *scene* '(scene-surface-of-revolution7)))
(setf (plane-area scene-plane1) 1.885327)
(setf (plane-area scene-plane2) 1.623139)
(setf (plane-area scene-plane3) 0.562917)
(setf (plane-area scene-plane4) 2.712013)
(setf (plane-area scene-plane5) 1.434931)
(setf (plane-area scene-plane6) 3.550574)
(setf (surface-of-revolution-area scene-surface-of-revolution7) 1.437068)
(img-adjacent 'scene-plane1 'scene-plane2)
(img-adjacent 'scene-plane3 'scene-plane4)
(img-adjacent 'scene-plane4 'scene-plane5)
(img-adjacent 'scene-plane6 'scene-surface-of-revolution7)
(scene-edge-info 'scene-plane3 'scene-plane4
 '((0.492970 4.415000 -2.858000)(1.038965 5.281000 -2.850000) 0.000128)
 '((0.492970 1.038965)(4.415000 5.281000)(-2.858000 -2.843000)))
(scene-edge-info 'scene-plane4 'scene-plane5
 '((1.116964 3.739000 -1.265000)(2.078956 4.764000 -0.725000) 0.001104)
 '((1.116964 2.104956)(3.739000 4.764000)(-1.265000 -0.703000)))
(scene-edge-info 'scene-plane6 'scene-surface-of-revolution7
 '((-2.003010 2.347000 0.526000)(-1.587013 2.161000 0.688000) 0.009401)
 '((-2.029010 -1.587013)(2.161000 2.385000)(0.497000 0.688000)))
(setq *scene-rows* 152)
(setq *scene-columns* 240)
(setq *scene-x-resolution* 0.026000)
(setq *scene-y-resolution* 0.040912)
; yres min 0.009000, max 0.259000, mean 0.040912, var 0.000061, n 12973
(setq *scene-point* '((99 9) (-2.965002 2.584000)))
(setq *scene-label-file* "/home/pixel/flynn/cad/imgs/occl8/occl8.label4.Z")
(setq *scene-image-file* "/home/pixel/flynn/cad/imgs/occl8/occl8.hips.Z")
```

Figure A.3: LISP description for the scene of Figure 4.8.

Bibliography

Bibliography

[1] G. Agin and T. Binford. Computer Description of Curved Objects. In *Proc. 3rd International Joint Conference on Artificial Intelligence*, pages 1113–1115, 1973.

[2] P. Allen and P. Michaelman. Acquisition and Interpretation of 3-D Sensor Data from Touch. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 33–40, 1989.

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

[4] A. H. Barr. Global and Local Deformations of Solid Primitives. In *ACM Computer Graphics SIGGRAPH '84*, pages 21-30, July 1984.

[5] H. R. Berenji and B. Khoshnevis. Use of Artificial Intelligence in Automated Process Planning. *Computers in Mechanical Engineering*, pages 47–55, September 1986.

[6] R. Bergevin and M. D. Levine. Generic Object Recognition: Building Coarse 3D Descriptions from Line Drawings. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 68–74, November 1989.

[7] P. J. Besl. Active Optical Range Imaging Sensors. In J. L. C. Sanz, editor, *Advances in Machine Vision: Architectures and Applications.* Springer-Verlag, 1988.

[8] P. J. Besl. Geometric Modeling and Computer Vision. *Proc. IEEE*, 76(8):936–958, August 1988.

[9] P. J. Besl and R. C. Jain. Three-Dimensional Object Recognition. *Computing Surveys*, 17(1):75–145, March 1985.

[10] P. J. Besl and R. C. Jain. Segmentation Through Variable-Order Surface Fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):167–192, 1988.

[11] B. Bhanu (ed.). Special Issue on CAD-Based Robot Vision. *COMPUTER*, August 1987.

[12] I. Biederman. Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 94(2):115–147, 1987.

[13] D. Blostein and N. Ahuja. Shape From Texture: Integrating Texture-Element Extraction and Surface Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1233–1251, December 1989.

[14] R. M. Bolle and D. Sabbah. Differential Geometry Applied to Least-Square Error Surface Approximations. In *Optical and Digital Pattern Recognition*, volume 754, pages 117–127. SPIE, 1987.

[15] R. C. Bolles and P. Horaud. 3DPO: A Three-Dimensional Part Orientation System. *International Journal of Robotics Research*, 5(3):3–26, Fall 1986.

[16] R. C. Bolles, P. Horaud, and M. J. Hannah. 3DPO: A Three-Dimensional Part Orientation System. *Proc. 8th Int. Joint Conf. on Artificial Intelligence*, pages 1116–1120, August 1983.

[17] T. E. Boult and A. D. Gross. On the Recovery of Superellipsoids. In *Proc. DARPA Image Understanding Workshop*, pages 1052–1063, 1988.

[18] J. Brady, N. Nandhakumar, and J. Aggarwal. Recent Progress in the Recognition of Objects from Range Data. In *Proc. 9th Int. Conf. on Pattern Recognition*, pages 85–92, November 1988.

[19] H. Brody. CAD Meets CAM. *High Technology*, pages 12–16, 1987.

[20] R. A. Brooks. Symbolic Reasoning Among 3-D Models and 2-D Images. *Artificial Intelligence*, 17(1-3):285–348, 1981.

[21] R. A. Brooks. Model-Based Three-Dimensional Interpretations of Two-Dimensional Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):140–150, March 1983.

[22] R. A. Brooks. *Model-Based Computer Vision.* UMI Research Press, 1984.

[23] C. M. Brown. Some Mathematical and Representational Aspects of Solid Modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(4):444–453, July 1981.

[24] C. M. Brown. PADL-2:A Technical Summary. *IEEE Computer Graphics and Applications*, 2:69–84, March 1982.

[25] M. S. Casale. Free-Form Solid Modeling with Trimmed Surface Patches. *IEEE Computer Graphics and Applications*, pages 33–43, January 1987.

[26] C. Chen and A. Kak. 3D-POLY: A Robot Vision System for Recognizing Objects in Occluded Environments. Technical Report TR-EE 88-48, School of Electrical Engineering, Purdue University, December 1988.

[27] C. Chen and A. Kak. A Robot Vision System for Recognizing 3-D Objects in Low-Order Polynomial Time. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(6):1535–1563, November/December 1989.

[28] S.-W. Chen. *3D Representation and Recognition Using Object Wings.* PhD thesis, Department of Computer Science, Michigan State University, 1989.

[29] S.-W. Chen and G. Stockman. Object Wings-2 1/2-D Primitives for 3-D Recognition. In *Proc. IEEE 1989 Conf. on Computer Vision and Pattern Recognition*, pages 535–540, June 1989.

[30] S.-W. Chen, G. Stockman, and N. Shrikhande. Computing a Pose Hypothesis from a Small Set of 3-D Object Features. Technical Report MSU-ENGR-87-001, Department of Computer Science, Michigan State University, 1987.

[31] R. T. Chin and C. R. Dyer. Model-Based Recognition in Robot Vision. *Computing Surveys*, 18(1):67–108, March 1986.

[32] J. Coggins and A. Jain. A Spatial Filtering Approach to Texture Analysis. *Pattern Recognition Letters*, 3:195–203, 1985.

[33] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide.* Society for Industrial and Applied Mathematics, 1979.

[34] S. A. Dudani, K. J. Breeding, and R. B. McGhee. Aircraft Identification by Moment Invariants. *IEEE Transactions on Computers*, C-26(1):39–46, 1977.

[35] D. Eggert and K. Bowyer. Computing the Orthographic Projection Aspect Graph of Solids of Revolution. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 102–108, November 1989.

[36] T.-J. Fan, G. Medioni, and R. Nevatia. Segmented Descriptions of 3-D Surfaces. *IEEE Journal of Robotics and Automation*, pages 527–538, December 1987.

[37] T.-J. Fan, G. Medioni, and R. Nevatia. Recognizing 3-D Objects Using Surface Descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, 1989.

[38] F. Ferrie, J. Lagarde, and R. Bajcsy. Darboux Frames, Snakes, and Super-Quadrics: Geometry from the Bottom-Up. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 170–176, 1989.

[39] P. J. Flynn and A. K. Jain. Surface Classification: Hypothesis Testing and Parameter Estimation. In *Proc. IEEE 1988 Conf. on Computer Vision and Pattern Recognition*, pages 261–267, June 1988.

[40] P. J. Flynn and A. K. Jain. CAD-Based Computer Vision: From CAD Models to Relational Graphs. In *Proc. IEEE 1989 Int. Conf. on Systems, Man, and Cybernetics*, pages 162–167, November 1989.

[41] P. J. Flynn and A. K. Jain. On Reliable Curvature Estimation. In *Proc. IEEE 1989 Conf. on Computer Vision and Pattern Recognition*, pages 110–116, June 1989.

[42] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

[43] Z. Gigus and J. Malik. Computing the Aspect Graph for Line Drawings of Polyhedral Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):113–122, February 1990.

[44] R. N. Goldman. Two Approaches to a Quadric Model for Surfaces. *IEEE Computer Graphics and Applications*, 3(1):21–24, September 1983.

[45] W. E. L. Grimson. The Combinatorics of Object Recognition in Cluttered Environments using Constrained Search. In *Proc. 1988 Int. Conf. on Computer Vision (ICCV '88)*, pages 218–227, 1988.

[46] W. E. L. Grimson. The Combinatorics of Heuristic Search Termination for Object Recognition in Cluttered Environments. Technical Report 1111, AI Laboratory, Massachusetts Institute of Technology, 1989.

[47] W. E. L. Grimson and T. Lozano-Pérez. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *International Journal of Robotics Research*, 3(3):3–35, Fall 1984.

[48] A. Gupta, L. Bogoni, and R. Bajcsy. Quantitative and Qualitative Measures for the Evaluation of the Superquadric Models. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 162–169, 1989.

[49] E. L. Hall, J. K. B. Tio, C. A. McPherson, and F. A. Sadjadi. Measuring Curved Surfaces for Robot Vision. *COMPUTER*, 15(12):42–54, December 1982.

[50] C. Hansen, N. Ayache, and F. Lustman. Towards Real-Time Trinocular Stereo. In *Proc 2nd Int. Conf. on Computer Vision*, pages 129–133, December 1988.

[51] C. Hansen and T. Henderson. CAGD-Based Computer Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1181–1193, November 1989.

[52] R. M. Haralick. Methodology for Experimental Computer Vision. In *Proc. 1989 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 437–438, June 1989.

[53] Y. C. Hecker and R. M. Bolle. Is Geometric Hashing a Hough Transform? Technical Report RC 15202(#67767), IBM Research Division, T.J. Watson Research Center, 1989.

[54] T. C. Henderson and C. Hansen. Multisensor Knowledge Systems. In A. K. Jain, editor, *Real-Time Object Measurement and Classification*, volume 42 of *NATO ASI Series*, pages 375–390. Springer-Verlag, Berlin, 1988.

[55] R. Hoffman, H. Keshavan, and F. Towfiq. CAD-Driven Machine Vision. *IEEE Trans. on Systems, Man, and Cybernetics*, pages 1477–1488, November/December 1989.

[56] R. L. Hoffman. *Object Recognition from Range Images.* PhD thesis, Department of Computer Science, Michigan State University, 1986.

[57] R. L. Hoffman and A. K. Jain. Segmentation and Classification of Range Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):608–620, September 1987.

[58] C. M. Hoffmann. *Geometric & Solid Modeling: An Introduction.* Morgan Kaufmann, 1989.

[59] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, Massachusetts, 1986.

[60] G. Hu and G. C. Stockman. 3-D Surface Solution Using Structured Light and Constraint Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:390–402, 1989.

[61] D. P. Huttenlocher and S. Ullman. Object Recognition Using Alignment. In *Proc. First International Conference on Computer Vision*, pages 102–111, 1987.

[62] K. Ikeuchi and T. Kanade. Automatic Generation of Object Recognition Programs. *Proc. IEEE*, 76(8):1016–1035, August 1988.

[63] S. Inokuchi, T. Nita, F. Matsuda, and Y. Sakurai. A Three Dimensional Edge-Region Operator for Range Pictures. In *Proc. 6th Int. Conf. on Pattern Recognition*, pages 918–920, 1982.

[64] D. J. Ittner and A. K. Jain. 3-D Surface Discrimination from Local Curvature Measures. In *Proc. IEEE 1985 Conf. on Computer Vision and Pattern Recognition*, pages 119–123, June 1985.

[65] A. Jain, T. Newman, and M. Goulish. Range-Intensity Histogram for Segmenting LADAR Images. *submitted to Pattern Recognition Letters*, 1990.

[66] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.

[67] A. K. Jain and R. L. Hoffman. Evidence-Based Recognition of 3-D Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):783–802, November 1988.

[68] R. C. Jain and A. K. Jain. Report: 1988 NSF Range Image Understanding Workshop. In R. C. Jain and A. K. Jain, editors, *Analysis and Interpretation of Range Images*, pages 1–31. Springer-Verlag, 1990.

[69] R. A. Jarvis. A Perspective on Range Finding Techniques for Computer Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):122–139, 1983.

[70] I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, New York, New York, 1986.

[71] A. Kak, A. Vayda, R. Cromwell, W. Kim, and C. Chen. Knowledge-based Robotics. *Int. J. Prod. Res.*, 26(5):707–734, 1988.

[72] M. R. Korn and C. R. Dyer. 3-D Multiview Object Representations for Model-Based Object Recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[73] E. Krotkov and J. Martin. Range From Focus. In *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, pages 1093–1098, April 1986.

[74] V. Kumar and L. N. Kanal. Parallel Branch-and-Bound Formulation for AND/OR Tree Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):768–788, 1984.

[75] Y. Lamdan and H. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Sceme. In *Proc. Second International Conference on Computer Vision*, pages 238–249, 1988.

[76] J. Z. Levin. Mathematical Models for Determining the Intersections of Quadric Surfaces. *Computer Graphics and Image Processing*, 11:73–87, 1979.

[77] W.-C. Lin and T.-W. Chen. CSG-Based Object Recognition Using Range Images. In *Proc. 9th Int. Conf. on Pattern Recognition*, pages 99–103, November 1988.

[78] D. G. Lowe. Three-Dimensional Object Recognition from Single Two-Dimensional Images. *Artificial Intelligence*, 31:355–395, 1987.

[79] H. Lu, L. G. Shapiro, and O. I. Camps. A Relational Pyramid Approach to View Class Determination. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 177–183, November 1989.

[80] A. Mitiche and J. Aggarwal. Detection of Edges Using Range Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):174–178, March 1983.

[81] D. G. Morgenthaler, K. D. Gremban, M. Nathan, and J. D. Bradstreet. The ITA Range Image Processing System. In *Proc. DARPA Image Understanding Workshop*, pages 127–142, 1987.

[82] M. E. Mortenson. *Geometric Modeling.* Wiley, New York, 1985.

[83] S. G. Nadabar and A. K. Jain. MRF Model-Based Segmentation of Range Images. *submitted to Second International Conference on Computer Vision*, 1990.

[84] V. S. Nalwa. Line-Drawing Interpretation: Bilateral Symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(10):1117–1120, October 1989.

[85] National Institute of Standards and Technology. Initial Graphics Exchange Specification, Version 4.0. Technical report, U.S. National Institute of Standards and Technology, 1988.

[86] R. Nevatia and T. Binford. Description and Recognition of Curved Objects. *Artificial Intelligence*, 8(1):77–98, 1977.

[87] D. Nitzan. Three-Dimensional Vision Structure for Robot Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):291–309, May 1988.

[88] J. O'Rourke and N. Badler. Decomposition of Three-Dimensional Objects into Spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1, July 1979.

[89] H. Park and O. Mitchell. CAD based planning and execution of inspection. In *Proc. 1988 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 858–863, June 1988.

[90] A. Pentland, T. Darrell, M. Turk, and W. Huang. A Simple, Real-Time Range Camera. In *Proc. 1989 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 256–261, 1989.

[91] M. Potmesil. Generating Models of Solid Objects by Matching 3D Surface Segments. In *Proc. 1983 Int. Jt. Conf. on Artificial Intelligence*, pages 1089–1093, August 1983.

[92] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. H. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, Cambridge, England, 1988.

[93] A. Requicha and H. Voelcker. Solid Modeling: A Historical Summary and Contemporary Assessment. *IEEE Computer Graphics and Applications*, 2(2):9–24, March 1982.

[94] A. Requicha and H. Voelcker. Solid Modeling: Current Status and Research Directions. *IEEE Computer Graphics and Applications*, 3(7):25–37, October 1983.

[95] A. A. Requicha. Representations for Rigid Solids: Theory, Methods, and Systems. *Computing Surveys*, 12(4):437–464, December 1980.

[96] D. F. Rogers. *Procedural Elements for Computer Graphics.* McGraw-Hill, 1985.

[97] D. F. Rogers and J. A. Adams. *Mathematical Elements for Computer Graphics.* McGraw-Hill, second edition, 1990.

[98] H. Samet. *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, 1990.

[99] R. Sampson, C. Swonger, and P. VanAtta. Real Time 3 Dimensional Image Processing for Robot Applications. Technical report, Environmental Research Institute of Michigan, 1984.

[100] P. T. Sander and S. W. Zucker. Tracing Surfaces for Surfacing Traces. In *Proc. First Int. Conf. on Computer Vision (ICCV '87)*, pages 241–249, 1987.

[101] L. E. Scales. *Introduction to Non-Linear Optimization.* Springer-Verlag, New York, New York, 1985.

[102] B. Smith, J. Boyle, J. Dongarra, B. Garbow, Y. Ikebe, V. Klema, and C. Moler. *Matrix Eigensystem Routines: EISPACK Guide.* Springer-Verlag, 1976.

[103] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):131–147, February 1990.

[104] T. Sripradisvarakul and R. Jain. Generating Aspect Graphs for Curved Objects. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 109–115, November 1989.

[105] M. Stephens, R. Blissett, D. Charnley, E. Sparks, and J. Pike. Outdoor Vehicle Navigation Using Passive 3D Vision. In *Proc. IEEE 1989 Conf. on Computer Vision and Pattern Recognition*, pages 556–562, 1989.

[106] C. V. Stewart and C. R. Dyer. The Trinocular General Support Algorithm: A Three-Camera Stereo Algorithm for Overcoming Binocular Matching Errors. In *Proc 2nd Int. Conf. on Computer Vision*, pages 134–138, December 1988.

[107] G. C. Stockman. A minimax algorithm better than alpha-beta? *Artificial Intelligence*, 12:179–196, 1979.

[108] G. C. Stockman. Object Recognition and Localization via Pose Clustering. *Computer Vision, Graphics, and Image Processing*, 40:361–387, 1987.

[109] G. C. Stockman, S. Kopstein, and S. Benett. Matching Images to Models for Registration and Object Detection via Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-4(3):229–241, 1982.

[110] Structural Dynamics Research Corporation. *Geomod 4.0 User Guide*, 1988.

[111] M. Subbarao. Direct Recovery of Depth-Map. In *Proc. IEEE Workshop on Computer Vision*, pages 58–65, 1987.

[112] S. Tanaka and A. C. Kak. A Rule-Based Approach to Binocular Stereopsis. In R. C. Jain and A. K. Jain, editors, *Analysis and Interpretation of Range Images*, pages 33–139. Springer-Verlag, 1990.

[113] Technical Arts Corporation. *100X 3-Dimensional Scanner: User's Manual and Application Programming Guide.* Redmond, Washington.

[114] W. Tiller. Rational B-Splines for Curve and Surface Representation. *IEEE Computer Graphics and Applications*, 3(6):61–69, September 1983.

[115] R. Vaillant and O. Faugeras. Using Occluding Contours for Recovering Shape Properties of Objects. In *Proc. IEEE Workshop on Interpretation of 3D Scenes*, pages 26–32, November 1989.

[116] J. Veenstra and N. Ahuja. Efficient Octree Generation from Silhouettes. In *Proc. IEEE 1986 Conf. on Computer Vision and Pattern Recognition*, pages 537–542, June 1986.

[117] B. Vemuri, A. Mitiche, and J. Aggarwal. Curvature-Based Representation of Objects from Range Data. *Image and Vision Computing*, 4(2):107–114, May 1986.

[118] B. C. Vemuri and J. K. Aggarwal. Resolving the Orientation and Identity of an Object From Range Data. In *Proc. 1987 Workshop on Spatial Reasoning and Multisensor Fusion*, pages 178–187, October 1987.

[119] J. Verly, R. Delanoy, and D. E. Dudgeon. Machine Intelligence Technology for Automatic Target Recognition. *The Lincoln Laboratory Journal*, 2(2):277–310, 1989.

[120] P. R. Wilson. A Short History of CAD Data Transfer Standards. *IEEE Computer Graphics and Applications*, pages 64–67, June 1987.

[121] A. Witkin. Recovering Surface Shape and Orientation from Texture. *Artificial Intelligence*, 17(1-3):17–45, August 1981.

[122] N. Yokoya and M. D. Levine. Range Image Segmentation Based on Differential Geometry: A Hybrid Approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):643–649, June 1989.