INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Finding and Matching Topographic Features in 3-D Object Meshes

Pamela J. Neal

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

1999

Program Authorized to Offer Degree: Electrical Engineering

In presenting this dissertation in partial fulfillment of the requirements for the Doctorial degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this thesis is allowable only for scholary purposes, consistant with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to University Microfilms, 1490 Eisenhower Place, P.O. Box 975, Ann Arbor, MI 48106, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature Pamela J Reed

Date July 27, 1999

University of Washington

Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Pamela J. Neal

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

Linda G. Shapiro

Reading Committee:

Eve A. Riskin

Jenq-Neng Hwang

Date: July 27, 1999

University of Washington

Abstract

Finding and Matching Topographic Features in 3-D Object Meshes

by Pamela J. Neal

Chair of Supervisory Committee

Professor Linda G. Shapiro
Department of Electrical Engineering

This dissertation defines a spatial symbolic model that can be used to describe classes of 3-D objects (anatomical and man-made) and a method for finding correspondences between the features of the symbolic models and point sets of 3-D mesh data. An abstract symbolic model is used to describe spatial object classes in terms of parts, boundaries, and spatial associations. A *working model* is a mechanism to link the symbolic model to geometric information found in a sensed instance of the class, represented by a 3D mesh data set. Matching is performed in a three-step procedure that first finds *working sets* of points in the mesh, then fits *constructed features* to these sets, and finally selects a subset of these constructed features that best correspond to the features of the working model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Praise God from whom all blessings flow! Faith in His intervention sees us through all the days of our lives.

I am deeply indebted to my advisor, Linda Shapiro. I thank her for her help, encouragement, and the time she spent with me developing the formal theory. I especially thank her for her patience and understanding about my tight time schedule. A student could not ask for a better mentor and advisor.

I wish to thank Zhenrong Qian for all her help in obtaining the object meshes. Her countless hours of work taking images and reconstructing the objects made it possible for me to finish my work in the short time I had available. I want to thank Habib Abi-Rached for the many theoretical discussions that helped enrich my understanding of the problem. I thank the Biological Structures Group for helping me find an initial direction for my research, and especially Cornelius Rosse for his enthusiasm and encouragement. I also want to thank the folks at the Vision and Autonomous Systems Center at Carnegie-Mellon University, for providing the mesh smoothing and normalization software.

I want to thank Donna Peterson for her support and encouragement over the past three years, and for being a true friend. I also wish to thank Alan Klayton for nominating me for the Air Force program under which I was supported, and his confidence in my success.

Finally, I want to thank my family. I thank my mother- and father-in-law, Carol and Curtis Neal for their support, and my children William and Kather-

ine for dealing with the "absentee mommy." I especially thank my husband Scott. Without his support, encouragement, and love, this research would not have been possible.

Chapter 1

OVERVIEW

1.1 Introduction and Motivation

Over the past few decades a great body of research has been devoted to three-dimensional object recognition. With the development of relatively cheap, accurate range sensors in the past few years, many researchers have moved from using intensity images to range data to take advantage of the added information. Intuitively it seems the three-dimensional data are a much closer representation of the physical world, so the recognition problem should be easier to solve, but time has shown this not to be so. Although the general problem of recognizing any object is still unsolved, many very successful techniques have been developed that handle particular classes of recognition problems. Identification of an object from a limited database of objects and finding exact pose of a geometric model from range data are two of the specific problems that are well understood and pretty much solved.

Our goal is to solve a slightly different problem: matching a symbolic 3D object model that represents a generic class of objects to sensed 3D data from an instance of that class. Even if the identity of the object is already known, identifying the occurrence of the features that are known to be present in such an object can be useful for locating the object in 3D space and for providing named landmarks to be used in further processing. Location and identification of such features can be used in a variety of applications, such as object reconstruction,

telerobotic operations, and medical informatics. If an object belongs to a certain class, say, "coffee cup" or "airplane," certain things are known about the object without seeing it. The coffee cup must have a handle, the bottom must be planar, the inside must be very concave, so there is need for a relatively thin rim around the top. The handle must join to the cup near the top and bottom. The list can go on, though we have still not described any particular coffee cup. Given a range image of a coffee cup, it would be useful to identify the inside, outside, rim and handle on that particular cup. The airplane must have two wings, the wings must be the same size. The airplane must be symmetric about a plane that divides the fuselage. It must have a tail. As with the cup, we have described no particular airplane, but given the range image of a particular airplane, it could be useful to determine what parts of the image correspond to the wings, what part corresponds to the fuselage, etc.

The goal of this dissertation is twofold: 1) to design a symbolic representation of three-dimensional objects that can be used to represent generic objects in a variety of applications, and 2) to design and implement a matching procedure that can find correspondences between symbolic model features and 3D mesh data. To accomplish these goals, we develop a new symbolic model for three-dimensional objects and use knowledge of an object in a new, two-step, bottom-up/top-down approach to assist in matching a three-dimensional mesh representation of the object to the symbolic model. This information can be useful for many applications, some of which are:

1. **More accurate and aesthetically pleasing reconstruction of an object.** There are several ways to obtain range data, the most common being active and passive stereo and laser range-finding. Though the data acquisition systems have improved in accuracy and affordability, the data obtained are still somewhat noisy and imperfect. These errors have a

variety of causes, including: 1) reflectance of the material from which the object is constructed, 2) specular highlights caused by type and placement of light sources, 3) difficulty in separating the background and the object, and (in the case of active methods) 4) features of the hardware scanner itself, such as beam width, scanning increment and so on. If object reconstruction is the goal, these range data are often used to generate a polygonal mesh. The method described in this thesis can be used to identify areas of the mesh that belong to certain features. The constraints on the model could then drive the object reconstruction, automatically correcting errors that occur in the mesh. For example, if an object such as a coffee cup is brought into a virtual world it can be rapidly scanned and a mesh fit to the noisy 3-D data. A generic mesh coffee cup model could be overlaid onto this imperfect mesh. The knowledge of the location of the rim, handle, bottom, etc. can be used for this overlay. The generic mesh could be deformed within certain constraints to fit the cup data without the inaccuracies present in the real data. More ambitiously, reconstruction of the cup could be done without a generic mesh model by placing some mathematical constraints on the features of the cup. Such constraints might include, "the top of the rim must lie in a plane parallel to the bottom of the cup," and "the walls of the cup must be of a uniform thickness," and "the body of the cup must be rotationally symmetric," and so on.

2. **Reconstruction from incomplete data.** Our method can be extended to reconstruct an object from only two or three carefully chosen views. The constraints from the knowledge-based model used in feature finding can also be used in the reconstruction. The features known by the model to be present in any mesh given to it will force the best-fit even on in-

complete data. If other features are not present, the system can form a reasonable hypothesis about the shape and location of those features based on the symbolic model. Even though such a reconstruction may not be completely accurate, it would at least be reasonable and useable.

3. **Identification of features in medical images.** As 3-D imaging methods in medicine become more widely used, the research into better visualization methods also increases. The obvious visualization method for techniques such as CT and MRI scans is a mesh representation of the anatomical objects found in the image sets. Reconstruction and diagnosis can be assisted by a method that "knows" where objects and features ought to be based on the location of those it does find. Since everyone's body is different, template matching to a canonical geometric model does not allow enough variation to be useful. A symbolic model that makes no geometric assumptions is needed.

4. **Recognition of objects in a limited environment.** While this system in its current form is not intended for object recognition, it should be fast enough so that it may be useful if the number of possible objects is small. In a manufacturing or sorting environment, where a rough mesh of a single object can be generated on the fly, the model may be useful to tell if an object is "A," "B," or "C," and where a robot should grasp the object.

1.2 Overview of the Proposed Methodology

This dissertation describes a new methodology for 3D feature recognition and a prototype computer system that illustrates the methodology. The first contribution of this dissertation is the definition of a 3D symbolic spatial model to represent generic object classes. The model was motivated by work in medical

informatics that is attempting to construct a symbolic model of the human body including all its anatomical structures [45][53]. The model we have proposed is in terms of volumes, surfaces, surface separators, and designated locations. The surfaces, separators, and locations correspond to the concept of *features* in computer vision.

The second contribution of this dissertation is a methodology for detecting the predefined features of a 3D instance of a generic model. The model is simply a data structure describing the features, their required properties, and the constraints on their spatial relationships. The data is a 3D mesh obtained from the mutual registration of several range images of a real 3D object. The methodology to be described in this dissertation is summarized in the block diagram of Figure 1.1. There are two stages; a bottom-up stage and a top-down stage, with the data input to the bottom-up stage, and model information input to the top-down stage.

The data are in the form of a triangular range mesh, which consists of vertices (3-dimensional sample points from the object) connected by edges to neighboring vertices. The first step of the methodology is to apply low-level classification and region-growing algorithms to the 3D mesh points to form regions of interest called **working sets**. Each working set identifies a portion of the mesh that could contain a feature of interest.

The working sets are input to the first step of the top-down stage; the *primitive fitting* step. All steps in the top-down stage of the methodology have input from the object model as well as from the previous step. During primitive fitting, the model supplies information about the type of primitives expected to be found and the expected parameter ranges for each primitive. These primitives are defined when building the model, and can be any function for which parameters of fit and a fit-error function can be defined. Each working set is fitted with all primitives using a best-fit algorithm. If the parameter range of the

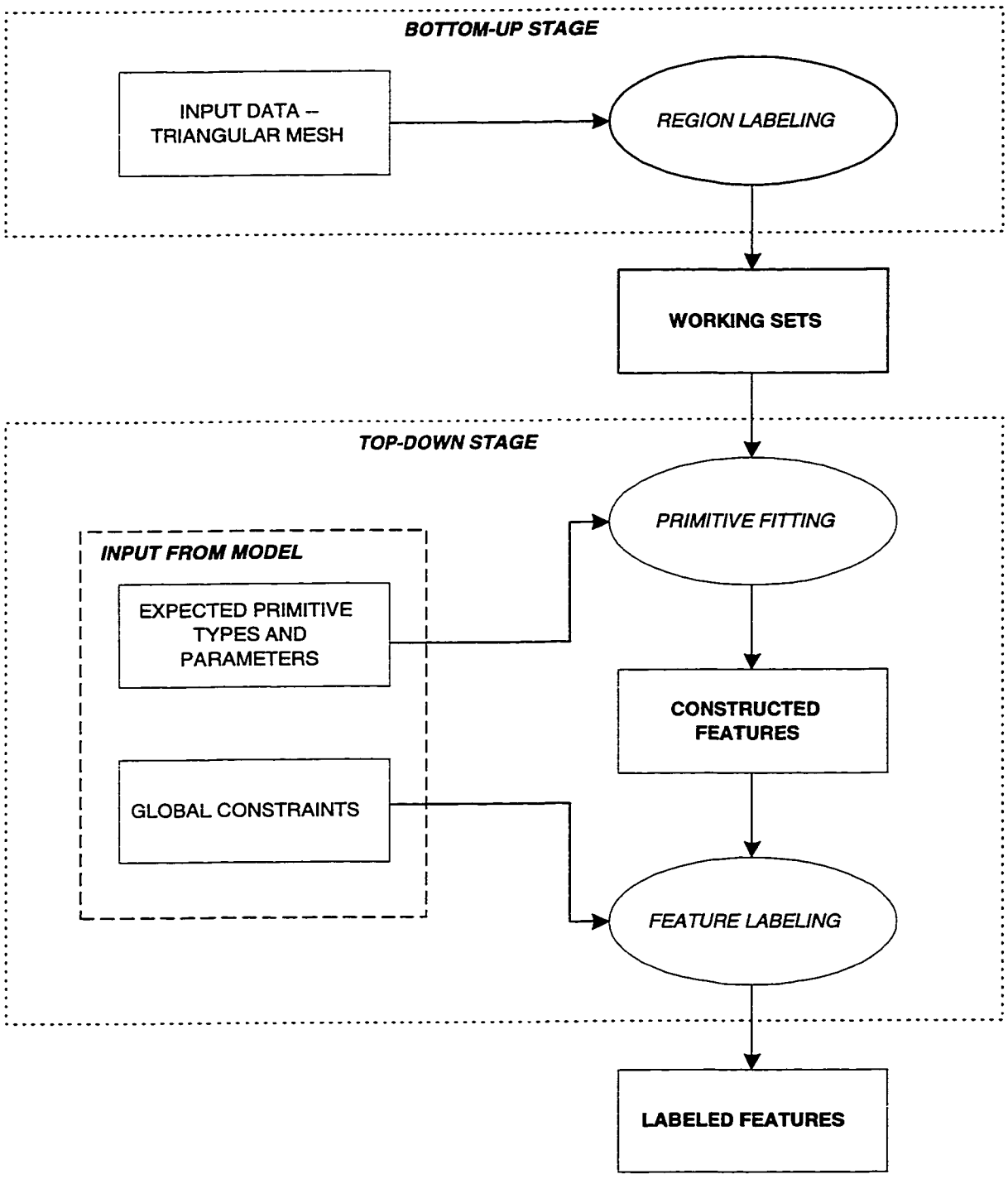


Figure 1.1: The proposed methodology for model-directed 3D feature extraction.

fitted primitive falls within the parameter range of a given feature, then that working set is labeled as a candidate for that feature. Each pair consisting of a working set and a potential feature label is called a **constructed feature**.

The constructed features are input to the last step in our methodology, the *feature labeling* step. The model supplies spatial constraints among the features to this step. These constraints are used to determine the best set of labels for the constructed features. If all model features are found in the data, then all will have matching unambiguous constructed features. If all model features are not found in the data, a list of unmatched features is produced as well as the labeling of matched features. In the event of more than one possible consistent labeling, a probabilistic function is used to determine the most likely correct labeling.

1.3 Background and Literature Review

Use of symbolic models of 3-D objects is often found in the field of artificial intelligence. The models are generally used in reasoning, and sometimes lack a real connection to a physically existing object [33]. Our model has two levels; a purely symbolic level that is useful for reasoning about an object and an intermediate level which contains physical constraints that enable us to tie real object data to the high-level model. In this way, we are able to use reasoning about the object to identify the real features in the mesh representation of the object.

In computer vision, parameterized models are most often the model of choice. Ideal parameters are stored in the model database. The range information is processed in a prescribed manner to produce similar parameters, then the data and model parameters are matched. This is a bit of an oversimplification of the process, but is sufficient to illustrate one of the main problems with this type

of representation: a different model must be generated and stored for each individual object desired to be recognized. In our methodology, only one model is needed for each *class* of object.

This literature review consists of two parts. The first part is a review of popular methods of object representation. The methods of object data and model representation are often the same and drive the method of matching. Our representations of the model and the object data are different from one another, and we believe this adds more flexibility to the system. We also believe that many of the current methods of representing objects could be incorporated into our model.

The second part is a review of the GRUFF system developed by Stark and Bowyer [57]. The GRUFF system uses a top down approach to recognize an object based on its possible function. While we are focusing on feature extraction rather than object recognition, this system is philosophically the closest to ours.

1.3.1 Object Representation

Volumetric Representations

Generalized cylinders. Generalized cylinders were developed by Binford [10] in 1971. Since then, much research has been done using generalized cylinders to model the parts of complex objects. Objects that are regular or have a natural axis of symmetry lend themselves to description by generalized cylinders. A generalized cylinder has an axis, a cross section, and a sweeping rule that describes how the cross-section evolves along the axis. In [17], Dion *et al.* use a single range image and first find the contours (step edges) in the image. They then extract axis points based on the contours and represent the axis points with a parametric equation of a curve in 3-D space. Due to the method

of axis extraction, the axis points become sparse at the junctions of parts, and so the equation fitting naturally stops at the junctions. They end up with a third degree polynomial representation for the axis of each part of the object. Equidistant cross sections are constructed for each part, and then the sweeping rule is estimated, again using third degree polynomials. This representation is sufficient for a decent reconstruction provided the contours and axes are adequately detected. A disadvantage of this method is that the axis extraction only works for elongated objects with parallel contours. Also, since there is no information about the object (this method was used to reconstruct, not recognize), only the visible data could be reconstructed. Objects as diverse as animals, vases, electric motors, airplanes, plumbing fittings, screws and buildings have been represented using generalized cylinders. Free-form objects are not well described by generalized cylinders, and unfortunately the generalized cylinder representation is not unique.

Superquadrics. Superquadrics were introduced to computer graphics by Barr [5] in 1981, and proposed for use as part primitives for computer vision by Pentland [47] in 1986. Superquadrics are a family of parametric shapes specified by a 3-D vector that sweeps out a surface parameterized in latitude and longitude in an object-centered coordinate system. There are three parameters to specify the size of the superquadric along the x, y, and z axes, and two relative shape parameters to specify the shape of the superquadric in the latitudinal and longitudinal directions. When all five parameters are unity, the superquadric is a unit sphere. This sphere is deformed into many shapes as the parameters change. In addition to the five parameters just described, three more parameters are needed to represent the translation of the superquadric from the origin, and another three to describe its orientation. This is a total of eleven parameters that must be estimated to completely specify a superquadric

at an arbitrary location and orientation. Much work has been directed toward finding the best way to estimate these parameters using range data, and several techniques have been developed over the past few years [2] [29] [56] [59] [44] [35] [63]. Still, the superquadric representation is not unique, and the systems that have the best results using superquadrics use them in conjunction with some other representation [25].

Geons. Geons, like superquadrics, are a class of generalized cylinder, but geon specification is less restrictive than superquadric specification. Geons are a relatively new representation, as they were introduced in 1987 by Biederman [9]. Geons are an attempt to represent an object in the way that humans initially recognize it. Biederman maintains that there are a small number of part primitives that can be combined to represent more complicated objects. Humans quickly and accurately identify the object by the Boolean combination and arrangement of those primitives. He calls these primitives "geons" (geometric ions) and there are 36 of them. They are classified on the basis of four shape attributes. Three of these describe the cross-section (shape, symmetry, and size), and the fourth describes the shape of the axis (straight or curved). Geons have been used off and on since their introduction to represent objects by parts. Bergevin and Levine [6] were first to make a vision system that used geons and the RBC (Recognition By Components) theory. They concluded that not everything could be represented by geons, but that a great many man-made objects could. They also concluded that the data must be properly segmented and the faces properly extracted and labeled before correct labeling of the geons could occur.

Later researchers used geons in combination with other representations to develop recognition systems. Raja and Jain [51] used both superquadrics and geons; superquadrics were used to represent the geometric data while geons

were used to classify the superquadrics into different shape categories. The idea was that if the superquadrics were classified into shape categories, indexing into the database would be faster. In later research, Raja and Jain [52] use a geon-like representation described by a Surface Adjacency Graph (SAG) to represent the parts of the range image. The SAG is a qualitative representation where the graph nodes are faces and a graph edge between two nodes means the surfaces are adjacent. The nodes and edges both have attributes; the node attribute is the type of surface while the edge has two attributes describing the angle between the surfaces it connects. This method of representation is very compact, but the results shown were on simple objects that could be unambiguously decomposed into the primitives. Dickinson *et al.* [15] did similar work, but used deformable models, and instead of a surface adjacency graph, they used a range aspect hierarchy. In a nutshell, this aspect hierarchy is a 4-deep tree structure with the modeling primitives at the top and aspects (views), faces, and boundary groups respectively at the next 3 levels. This allows representation of all views of the primitives without redundancy. The geon representation is a rough model of the object and is used for basic categorization of an object. It does not represent surface details. (This can be said of all the generalized cylinder representations). Still, after a decade of research using geons, many researchers feel they are a good representation for a generic object recognition system [16].

Surface Patches Representations

In contrast to volumetric representations, representation by surface patches is concerned with the local surface characteristics of the object. In general, surface patches are generated for an object and then similar adjacent patches are merged to form a set of surfaces that can be matched to a model of the ob-

ject. The most common surface patch representations are planar patches and quadric patches; higher order equations are more difficult and much more time-consuming to estimate and do not contribute much additional information for many types of objects. Usually shape information about the patches is desired, so parametric surfaces are used and differential geometry techniques applied in order to get the principal curvatures of the region. The principal curvatures are just the maximum and minimum curvatures of a surface, while the principal directions are the directions in which those curvatures occur. More useful measures of curvature are the mean (H) and Gaussian (K) curvatures which are derived from the principal curvatures. Surfaces can be divided into eight basic types: peak, ridge, saddle ridge, flat, minimal (like a saddle ridge, only flatter), pit, valley, and saddle valley based on the sign of the H and K curvatures. Many researchers use this surface classification scheme. An excellent review of differential geometry and the useful properties of H, K curvatures can be found in [7].

Signature Representations

The representations above are good for describing certain classes of objects, but they are not suitable to represent the class of objects called "free-form." Free-form objects are those which (generally) lack symmetry and are too complex to be represented by simple planar or quadric equations. Several representations for free-form objects have been developed recently. Due to the nature of free-form objects, these representations do not do any segmentation of the image.

Point signatures. Chua and Jarvis [12] developed a representation for free-form objects that they called "point signatures." The point signature of any given point in a range image is formed by placing a sphere of some radius centered at the point. This sphere will intersect the object surface forming a

3-D space curve. A reference frame is established based on a planar fit to the intersection curve. That plane is translated to the original point, and a signed-distance profile is formed using the original curve and the projection of the curve onto the translated plane. This point signature can be represented as a simple 1D signature pattern and can be rapidly matched to the model point signatures in the database. The point signature of the 3-D model is calculated for every point on the model, but this can be done off-line. The max and min values of the point signatures are used as indices into the database, and this allows for rapid retrieval of candidate models using the point signatures of the image. This representation uses local surface shape information only, and does not attempt to characterize the surfaces of the object.

Shape Spectrum. While point signatures exploit local surface characteristics only, Dorai and Jain [18] integrate local and global shape information in the representation scheme they call COSMOS (Curvedness-Orientation-Shape Map On Sphere). COSMOS uses a concept called the shape spectrum to group shape categories of free-form surfaces together. The shape spectrum uses the principal curvatures as described above, but instead of using the sign of the Gaussian and mean curvatures to determine shape type, they define a new, continuous measure called shape spectrum. This allows classification on a smooth scale so the standard shape types smoothly transform from one type to the next. In this manner, more subtle shape variations can be described. They also add curvature information to the representation so scale differences may be captured. The object is described in terms of surface patches that are different shapes, which the authors call the Constant-Shape Maximal Patch (CSMP). These surface patches are mapped onto a Gaussian sphere, and additionally a region adjacency graph is formed which keeps track of the connectivity of the patches. In further work [19] and [20] they go on to use the representation in

matching and recognition.

Splashes. Stein and Medioni [55] used a combination of representations to model free-form objects. They used *supersegments* to represent 3-D curves and *splashes* to represent those parts of a free-form object that are not detected by a curve extractor. Supersegments are simply a polygonal approximation to the 3-D curve with information such as number of segments and the angles (both curvature and torsion) between segments. Splashes are a type of small surface patch representation derived from each point on the surface. The surface normal is determined at the point, then a circular slice with some geodesic radius is computed. A surface normal is then computed at every point on that circle (actually at fixed angular intervals). The representation is based on the interval angle and the angle between the original point and the surface normal on the circle at that interval angle. Functions of these two angles are separately mapped to a local coordinate frame for that point. The two mappings can be combined into a 3 dimensional vector mapping. This mapping forms a 3-D curve which can be approximated in the same manner as the supersegments. Now both the edges and the free-form surface are represented with the supersegments, so they can be encoded and matched in the same manner. The encoded supersegments serve as keys into a table for matching purposes.

Spin Images. Recently, Johnson and Herbert [39] have developed a representation for 3-D objects based on oriented points.[1] The oriented point is used to define a 2-D basis using the tangent plane and the normal to the point. A *spin-map* is a function that maps 3-D points to the 2-D basis of a particular oriented point. They call it a spin-map because the basis can spin on its axis with no effect on the coordinates of points with respect to the basis. They then gen-

[1]An oriented point is a three dimensional point with an associated direction.

erate a *spin-image* by applying the spin-map to the 3-D points on the object. If a spin-image is generated for every point on the model, then a complete, unique description of the object is made available. In further work [40], they used a mesh model of the object to generate the spin images and obtained very good results matching the spin images from the model to the spin images generated from the range data, even in the presence of clutter and occlusion.

The object representations described in the above paragraphs have been fairly successful in their own domain. A drawback of them all is that the exact model of the object being matched must be available to generate the model for the algorithm. This issue is avoided in some cases by training the systems on many range images of the object it will later be asked to recognize, but they still must use the very object they want to find. This works well in a controlled environment, but we wish to use a more general object model.

1.3.2 Object Recognition Using Form and Function.

Perhaps the work most related to ours in approach is that of Stark and Bowyer [57]. In the GRUFF (Generic Recognition Using Form and Function) system, Stark and Bowyer use range images as input data and match various parts of the data to a symbolic model. The method of object recognition is to determine if an object could function as one of the known objects in the database. GRUFF uses three types of knowledge to accomplish this task. *Knowledge primitives* are a small set (six, at this time) of parameterized procedures designed to evaluate an object or parts of an object to see how well it fits the parameters. For example, the *dimensions* primitive takes as arguments a functional element (a portion of the structure that provides a required function), dimension type (such as width, depth, height, etc.), and range parameters, and returns a measure between 0 and 1 that tells how well the functional element fits into the range parameters.

The second type of knowledge is the *functional property*. Functional properties are symbolic labels that can be attached to an object or part of an object. They indicate that the object satisfies a requirement particular to the function of a given category of object. Whether or not an object has a particular functional property is determined by a sequence of calls to knowledge primitives. There are many functional properties defined in the GRUFF system. Examples for an object such as a chair are; *provides sittable surface* and *provides stable support*.

The last type of knowledge is the *category definition tree*. The nodes of this tree are the category labels arranged in a hierarchy defining sub- and superordinate relationships. The leaves of the tree specify the functional properties relevant to each category. For example, to find the description of a cup the superordinate category in the tree would be **dishes**, which has a subcategory **cup/glass**, which has a subcategory **mug with handle**. Each category has its own functional properties which are inherited by all subcategories.

In the analysis process, GRUFF does some low-level preprocessing and then uses a top down approach to determine if an input object can function as one of the objects present in the category definition tree. During the preprocessing stage all potential functional elements are labeled, and the object structure is evaluated using simple measures (called *key properties*) such as volume, center of mass and convex hull. All categories and subcategories have a key property and a range of values associated with that key property. These measures are used to quickly eliminate impossible categories. For example, if a glass were input to the system the main category *furniture* can be pruned from the tree right away just based on surface area. After impossible categories are

eliminated, the remainder of the categories and subcategories are indexed according to how close the key value falls to the center of the range for that category. Once the indexing is done, the associated categories and subcategories are ranked highest to lowest and depth-first traversal of the category definition tree is performed starting with the highest ranked category.

During the traversal of the category definition tree, evidence of the object fitting each category is accumulated (via an evaluation measure) as the functional properties are scored according to the evaluation of the required knowledge primitives. If a functional evaluation is failed, that part of the tree is no longer considered. The better the score on the functional element evaluation, the higher the evaluation measure. Also, the deeper the evaluation goes into the tree, the higher the evaluation measure. The tree is only searched until the input object is found to fulfill the functional requirements of some category with an evaluation measure of 0.4 or greater (the evaluation measure is a number between 0 and 1). Since it is possible that a category might exist with a lower indexed rank but a higher evaluation measure, it is an option to continue the processing until all potential recognition results are found.

This work used symbolic models, knowledge of the functionality, and a top-down strategy to match the input object to the symbolic model. Although we are not concerned with functionality or recognition of an entire object, we do use a symbolic model, knowledge of the expected features in the object, and a top down strategy for object-model matching.

1.4 Overview of Thesis

The motivation for using a triangular mesh representation and our method of obtaining a mesh for an object are discussed in Chapter 2. In Chapter 3 we develop the formal definition of the model and discuss the feature detection algorithm in detail. Chapter 4 discusses our prototype system developed using the model and algorithm defined in Chapter 3. Chapter 5 describes the models used in our experiments. In Chapter 6 we present a set of experiments using our system and discuss the results. A summary and future work are discussed in Chapter 7.

1.5 Research Contributions

The principal research contributions of this thesis are:

- Development of a new way of modeling 3-D objects. This new method consists of a high-level symbolic model coupled with an intermediate model. The intermediate model contains quantitative parameters that can be compared to actual values obtained by fitting specified functions to the range data.

- Development of a two step model-directed search procedure that takes advantage of both bottom-up and top-down techniques.

- Introduction of *constructed features* which have a structural foundation in the data but can exploit local relationships between groups of points having similar characteristics.

Chapter 2

DATA REPRESENTATION AND ACQUISITION

2.1 Introduction

There are several ways commonly used to represent range data. At the very lowest level is simply the points themselves, as obtained from the data acquisition system. Many of the systems ([34] [17] [18] [12]) described in the first chapter operate directly on those raw data. One notable exception is Johnson [39], who processes the data into a **triangular mesh** before applying his spin image techniques to it.

In this chapter we first discuss our motivation for the use of triangular meshes, provide a definition of meshes, and state our assumptions about the meshes that we use. We then discuss our method of range data acquisition and go into some detail on our adaptation of *spacetime analysis* [13] to active stereo. Next is a brief discussion of the range data registration techniques and the method for generating a rough mesh. Finally, we briefly discuss the tools used to smooth and normalize the mesh.

2.2 Triangular Meshes

2.2.1 Motivation for use of Triangular Meshes

Object representation using triangular meshes has been a long standing technique used by computer graphics researchers in the effort to reproduce objects in a visually pleasing manner. Computer vision researchers have not used this

representation in the past due to computational and storage concerns. In recent years, vast improvements in both processing power and storage capabilities have overcome these issues. Besl [8] discusses the advantages and weaknesses of using triangular meshes as a primary representation, and strongly advocates their use in computer vision. Triangular meshes provide both a sampling of and approximation to the underlying surface, and connectivity information between the sampled points. Techniques used to produce the mesh will vary depending on the source of the data. Sources of data can be very diverse, including ranging equipment (which produce 3D points), medical imaging equipment (which produce images from which extracted contours lead to 3D), and CAD/CAM systems (which produce NURBS surfaces). A method such as ours that uses a triangular mesh as an input will not be dependent on any particular type of equipment and can use input from a variety of sources.

2.2.2 Definition and Assumptions

Foley et al. [28] defines a polygonal mesh as "a collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons." A triangular mesh is just a polygonal mesh with the added requirement that the polygons are all triangles. The input to our method is assumed to be a simple (edges can be adjacent to no more than two faces) triangular mesh. Most surface reconstruction algorithms create simple meshes, and if the mesh is not triangular it is fairly trivial to break each planar polygonal face into multiple triangular faces. We further assume that the edges are normalized, that is, the distance between vertices is fairly regular throughout the mesh. Since many meshes do not satisfy this assumption, we use tools developed by Johnson [38] to normalize the input meshes.

2.3 Range Image Acquisition

Structured light is a commonly used technique for obtaining range information about a scene or object [3]. By projecting structured light onto a surface, stereo matching can be accomplished on even a blank, featureless surface to obtain depth information. Light striping is a form of structured lighting, and consists of scanning a narrow sheet of light across an object, and taking a snapshot with a pair of cameras each time the sheet of light is incremented. Stereo matching and triangulation are then performed on the pairs of images to obtain depth information. The matching task is much easier and more reliable using these thin sheets of light, but because the light projection tends to be more than a pixel wide (on the order of ten pixels, for our system) errors in matching can still occur and give erroneous depth information. Use of an optical triangulation system is another way to obtain range data, often more accurate than the light striping system. Spacetime analysis was introduced as a technique to reduce errors in range data obtained with optical triangulation systems, specifically in the case of a laser range scanner [13]. Many of the errors associated with the laser range scanner are also present in the light striping system we use. The spacetime analysis technique should work equally well in reducing those errors in our light striping system. In this section we extend the technique of spacetime analysis to an active stereo vision system (specifically a light striping system) and demonstrate how spacetime analysis coupled with multiple cameras can provide more accurate range information. First, we review the spacetime analysis technique as developed by Curless and Levoy, and discuss why it should work for our system as well. We then give a brief discussion of our decision to use four cameras for our stereo system instead of two. Next, we describe our system and discuss the data acquisition and some of the spacetime implementation details. Finally, we compare our results using the

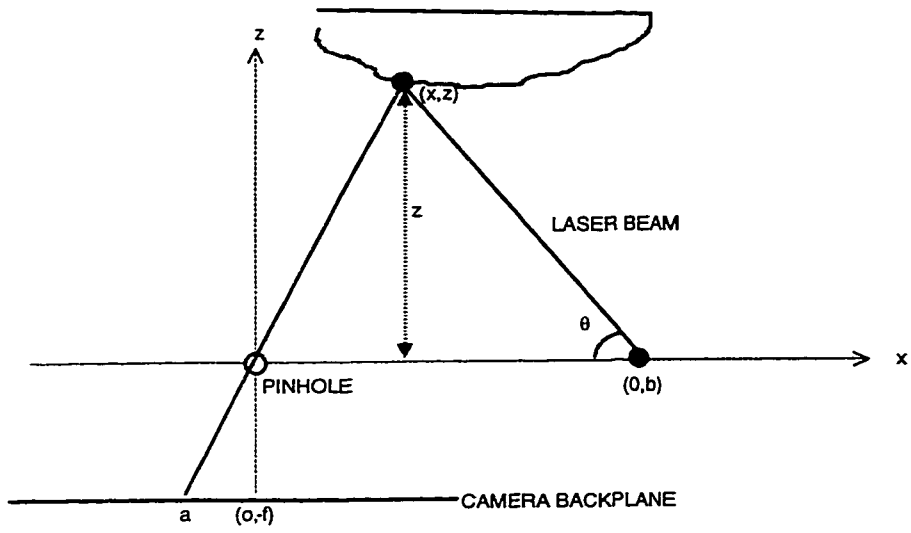


Figure 2.1: Optical triangulation geometry in 2-D. The camera, object and laser position are all fixed, and the positions are known. The laser angle, θ, is varied to sweep the spot over the surface. The coordinates (x, z) can be calculated by simple trigonometry.

modified spacetime method vs. traditional methods of range data calculations.

2.3.1 Spacetime Analysis - Laser Scanner

Curless and Levoy introduced Spacetime Analysis as a method to improve the accuracy of an optical triangulation system. A 2-D version of an optical triangulation system is shown in Figure 2.1.

The camera, object and laser position are all fixed, and the positions are known. The laser angle, θ, is varied to sweep the spot over the surface. The coordinates (x, z) can be calculated by simple trigonometry: $\tan\theta = z/(b - x)$ and $z/x = f/(-a)$. In real systems, the laser spot has a finite diameter, so the image of the spot has finite width. To calculate the depth value, the center of the imaged spot must be calculated. The intensity of the laser spot is generally assumed to be Gaussian, so typically researchers use mean, median or

peak of imaged light as the center. This gives correct results if the surface is perfectly flat and featureless, but becomes less and less accurate as the surface features vary sharply. Errors are introduced due to perturbations in the shape of the laser spot (or illuminant), causing the peak of the illuminant to be off-center, thus giving erroneous range information. The perturbations can be caused by a variety of characteristics of the object's surface, including reflectance variances, geometric deviations (from planar), and partial occlusion of the light paths. Figure 2.2 (taken directly from Curless and Levoy's paper) shows examples of some of these errors. Spacetime analysis is a new method of calculating the center of the laser spot image. Conceptually, instead of looking at the whole imaged spot and trying to find the center, they look at each individual pixel and find the time of its peak illumination. Since the illuminant has a Gaussian shape, as the image spot moves across the backplane a given pixel will first see the tail of the illuminant, increasing in a Gaussian fashion to the peak, and then tapering off at the other tail. The time of the peak is recorded; the intensity is no longer important for range calculations. The range information for that pixel can now be calculated from the angle of the laser at the time the pixel was at peak intensity. Results were given comparing the traditional method to the spacetime method on some problem images, and the results looked promising.

2.3.2 Spacetime Analysis - Applied To Active Stereo

The system we use is not a laser scanning system, but an active light-striping system. A narrow beam of light is scanned across the object while multiple cameras take images. Though the systems are different, our technique is still prone to the same types of errors from the same causes as the laser scanning system. Figure 2.3 shows an example of error caused by a corner. Stereo matching is done for each stripe of light. During the matching process, the pixels where

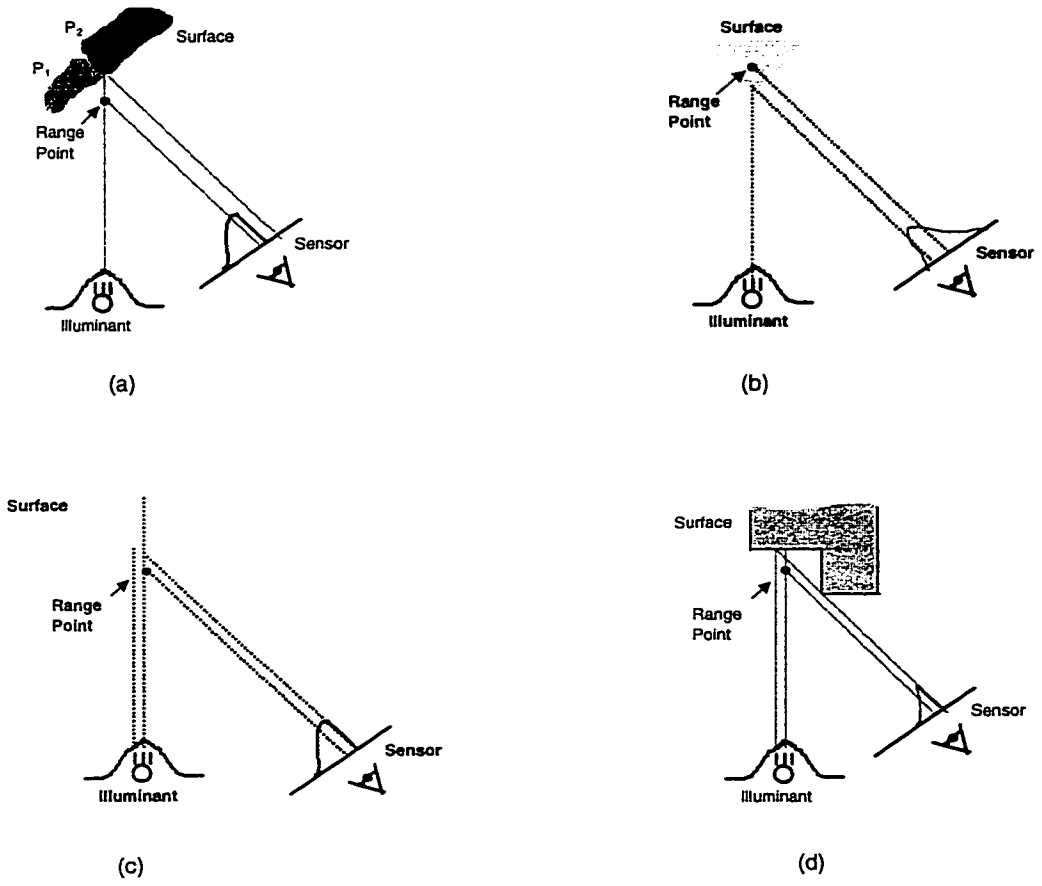


Figure 2.2: Range errors using traditional triangulation methods. (a) Reflectance discontinuity. (b) Corner. (c) Shape discontinuity with respect to the illumination. (d) Sensor occlusion.

the average peak intensity occurs are matched. In the example shown in Figure 2.3, matching the average peak intensity will result in an incorrect depth value. If only intensities that were the same were matched, the depth value would still be incorrect due to the partial occlusion of the illuminant caused by the corner. Since the errors in our system have the same root cause as those in the laser scanning system, spacetime analysis can be extended to reduce the errors. Instead of using spacetime analysis to find the proper angle, we use it to find the proper correspondence of pixels between two stereo images. A very thin vertical sheet of light is scanned across an object, and a snapshot is taken each time the position of the sheet of light is incremented. The time when a given pixel is at peak intensity is recorded and assigned to that pixel as its value. After the entire image is scanned, all pixels that saw any light have a time value associated with them.

Figure 2.4 illustrates the epipolar geometry used to match pixels in a stereo pair of images. The method is to take a pixel, P_0, in one image, cast a ray from P_0 through the optical center of the camera, and form the epipolar plane defined by C_0 (the optical center of the first camera), C_1 (the optical center of the second camera), and P_0. The intersection of this epipolar plane and the image plane of the second camera forms a line, ep_1 [23]. The pixels intersected by ep_1 are searched, and a match is found if there is a pixel P_1 with a time value corresponding to the time value of pixel P_0. Since the sheet of light used to generate the time values is roughly orthogonal to the epipolar plane, there can be at most one pixel along the epipolar line that has a matching time value. Due to occlusion there may be no pixel that has a matching time value. The example shown in Figure 2.3 is one such instance of no matching time value. Thus, instead of an erroneous depth value, we get no depth value for that particular point. This is a correct result since that point is occluded from the second camera. If a match is found, a depth value is calculated using standard triangulation

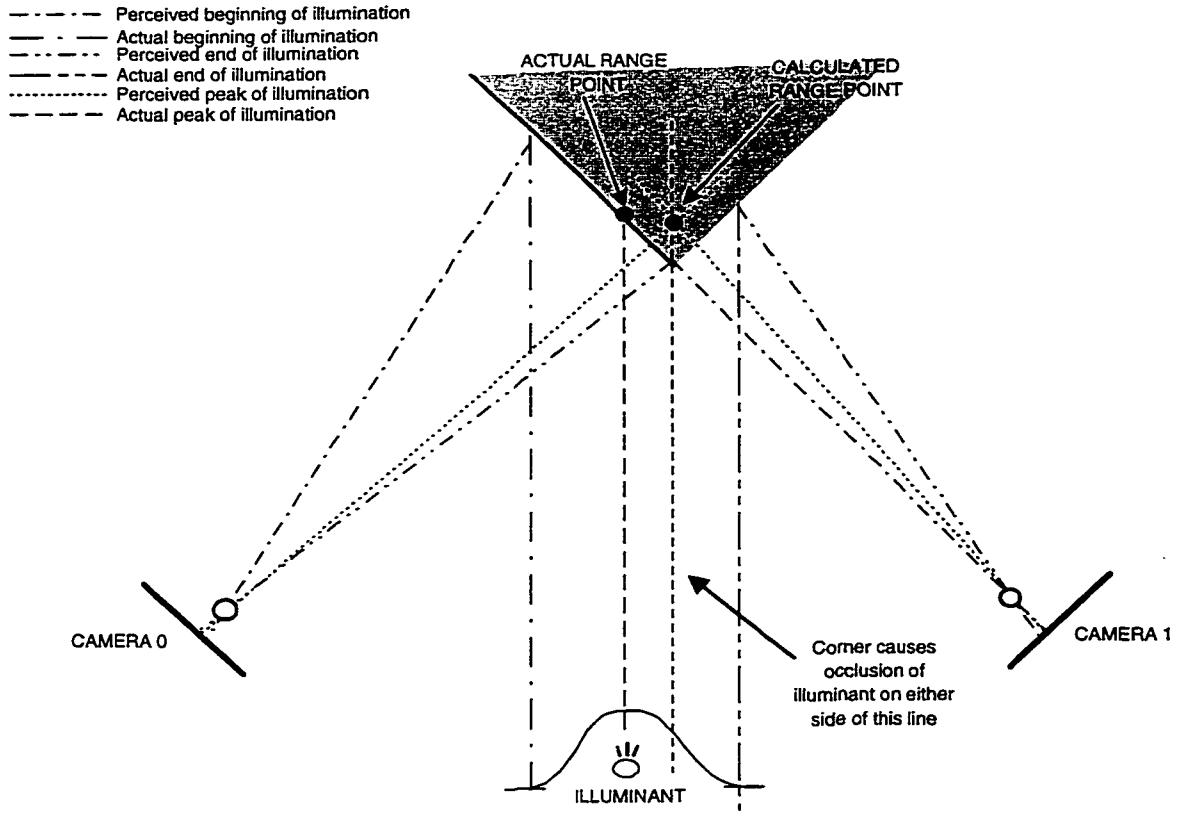


Figure 2.3: Range error in light striping system. Both cameras should map the center of the Gaussian intensity peak to the same pixel. The corner causes all illumination right of the dashed line to be occluded from camera 0, and all illumination left of the dashed line to be occluded from camera 1. This causes an erroneous depth calculation.

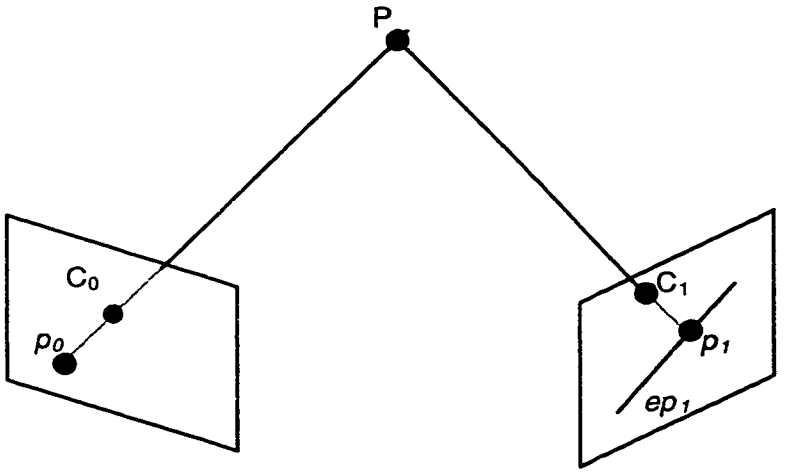


Figure 2.4: Epipolar geometry. p_0 is the image of point P. C_0, C_1 are the optical centers of cameras 0 and 1. The corresponding image point p_1 in camera 1 must lie on the line ep_1, which is the image of a ray cast from p_0 through C_0.

methods.

2.3.3 Multiple-Baseline Stereo With Spacetime Analysis

Multiple-baseline stereo is just using more than two cameras in known relation to each other so more images are available for stereo matching. Often [61][46], multi-baseline is used in passive systems or as the "active" part of an active stereo system (a single camera is used, and moved a known amount to obtain multiple images). Our primary motivation for using all four cameras is to reduce the problems of occlusion, as illustrated in Figure 2.5.

System Setup And Data Acquisition

Physical Setup. Our data acquisition system consists of four Sony DXC107A CCD cameras mounted on a single horizontal bar and a slide projector mounted on a translation table. A solid slide with a very thin vertical cut in the center is used to project a light stripe onto the object. The translation table and frame

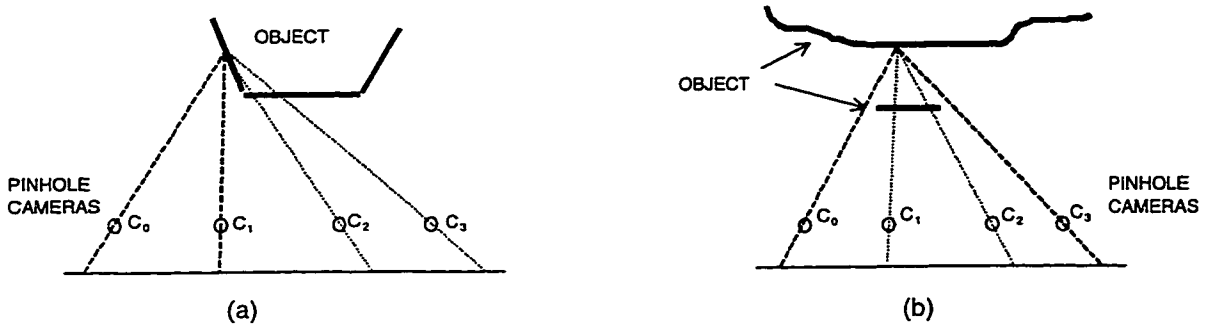


Figure 2.5: Reducing occlusion by using four cameras. (a) If only cameras 0 and 3 were used, data would be lost. (b) If only cameras 0 and 1 (or 2) were used, data would be lost.

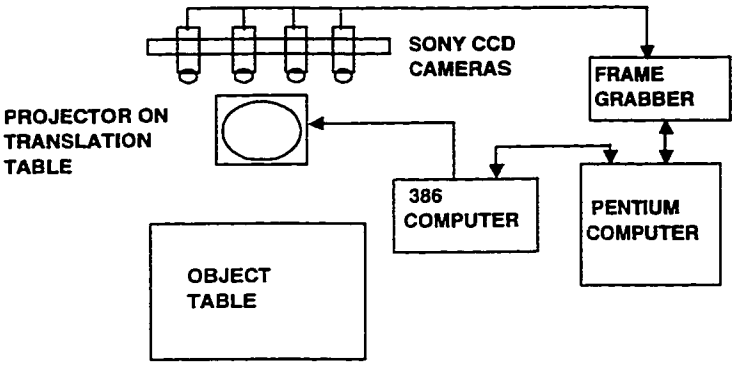


Figure 2.6: Physical setup of data acquisition system.

grabbers for the cameras are computer controlled and synchronized so the time of the frame and step size of the table are known and adjustable (see Figure 2.6). The 386 computer controls the stepper motors that move the translation table. The Pentium computer receives the images from the frame grabber and preprocesses the data. The 386 and Pentium have a communication link, and the Pentium controls timing of the translation table movement so the preprocessing can finish before the light stripe is moved.

Preprocessing. The cameras are calibrated using the Tsai method [61], a common camera calibration technique. As the light stripe is moved across the object, all illuminated pixels are observed. For each illuminated pixel, peak intensity is calculated. This time of peak intensity is calculated using the weighted average:

$$t_{I_{peak}} = \frac{\sum t_i I(t_i)}{\sum I(t_i)}$$

where t_i is the time at which the intensity is being measured and $I(t_i)$ is the intensity at that time. The range of i will be determined by the selected step size and beam width; it is not known *a priori*. The time of peak intensity, as well as a σ value is stored for each pixel. This σ is the standard deviation, in time, about the peak (see Figure 2.7):

$$\sigma^2 = \frac{\sum (t_{I_{peak}} - t_i)^2 I(t_i)}{\sum I(t_i)}$$

This value will be necessary in the matching process since the time of the peak is a calculated value and will not likely be exactly the same for the same pixel in two different cameras. Also, as shown in Figure 2.7, we discovered "false peaks" may occur due to inter-reflections on a shiny object. These are removed by temporarily storing a peak, then comparing it to a new peak that might occur. The method works well as long as the intensity value of the pixel goes to zero between peak occurrences. The final result of the preprocessing is two arrays for each camera; one holding time values and one holding σ values at each pixel. Because the time values can be calculated "on the fly," we do not have to store the entire space-time image as Curless and Levoy did. This greatly reduces the size of our storage needs. These files are then transferred to a Silicon Graphics O2 for data reduction.

Data Reduction. In the data reduction phase, the 3-D points are reconstructed from the arrays produced in the preprocessing phase. One camera

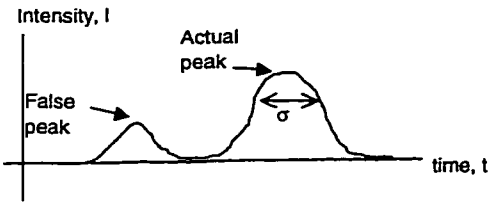


Figure 2.7: Peak intensity for a pixel. Time of actual peak intensity for each pixel is stored. False peaks are caused by inter-reflections, and must be removed.

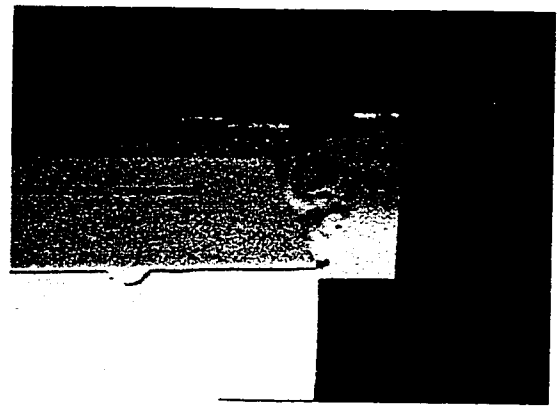
is selected as the reference camera. For each pixel that has a time value (background pixels have no time value associated with them), the epipolar line is projected into each of the other three cameras. The epipolar line is searched for a pixel with a time that matches within σ of the reference pixel. If a match is found, a depth point is then calculated using standard triangulation methods. In this manner, up to three depth points for each 3-D point may be found.

We experimented with several different combinations of the available depth points, to see which gave the best results. First, we just averaged the points that were available for each 3-D point. In other words, if a match occurred (from the reference camera) in all 3 other cameras, we averaged the three points together. If we had a match in 2 other cameras, we averaged the two points. And finally, if the point matched in just one other camera, we reconstructed it. This method gave a very dense cloud of points, but was also quite noisy. We then tried reconstructing a point only if it were matched in all 3 cameras. The results were better, but now the point cloud was, of course, much less dense and we lost some detail. We also still had a lot of noise, so we developed a noise test. This test just calculated the Euclidean distance between the points to be averaged. If this distance was larger than a certain threshold, the points were discarded and the reconstruction was not done for that particular point.

The threshold was determined heuristically, but could be developed in a more rigorous manner such as the one used by Stockman, *et. al* [58]. The noise test significantly improved the result, but naturally the point density was lower when using this test.

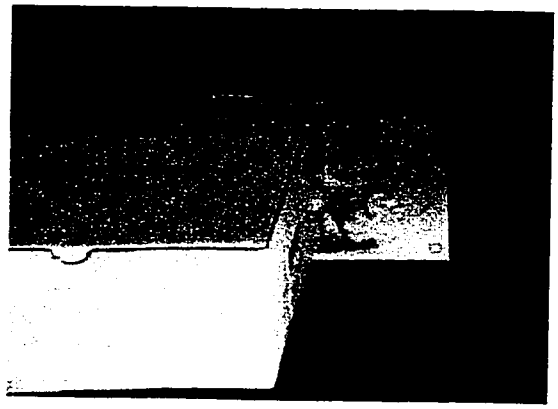
Results of using the spacetime method We observed two main improvements to our acquired data by using spacetime analysis over traditional triangulation methods. First, we saw an improvement in accuracy of reconstructed points. To illustrate this improvement, we use two boxes. The intensity image of the boxes from all four cameras is shown in Figure 2.8. For purposes of illustration, the reconstructions in this example are done using only two cameras, cameras 0 and 3. The top image in Figure 2.9 shows the reconstruction done using the traditional method. There are many reconstructed points that appear to belong to the right face of the near box. However, we can see from Figure 2.8 that the right face of that box appears in camera 3 but is occluded from camera 0. Those points in the reconstruction must have come from false matches since the data are simply not available to construct that face. Also notice the lettering on the top of the near box. Since that lettering is black (in very high contrast to the light-colored box), the standard method of matching the pixels was affected in the manner shown in Fig. 2.2(a). The bottom image in Figure 2.9 is the reconstruction using the spacetime method. The top of the near box is reconstructed correctly (no words apparent), and the right side of the near box has no reconstructed points.

The second major improvement was in the effects of inter-reflection on the reconstructed data points. Figure 2.10 illustrates those effects. The object used was just a plain piece of paper with contrasting horizontal and vertical colorbars. The paper was folded, then unfolded at junction of the black and white colorbars, so a slight crease was present at the junction. Figure 2.10(a)



(a) View from Camera 0

(b) View from Camera 1



(c) View from Camera 2

(d) View from Camera 3

Figure 2.8: Views of 2 boxes as seen from 4 cameras.

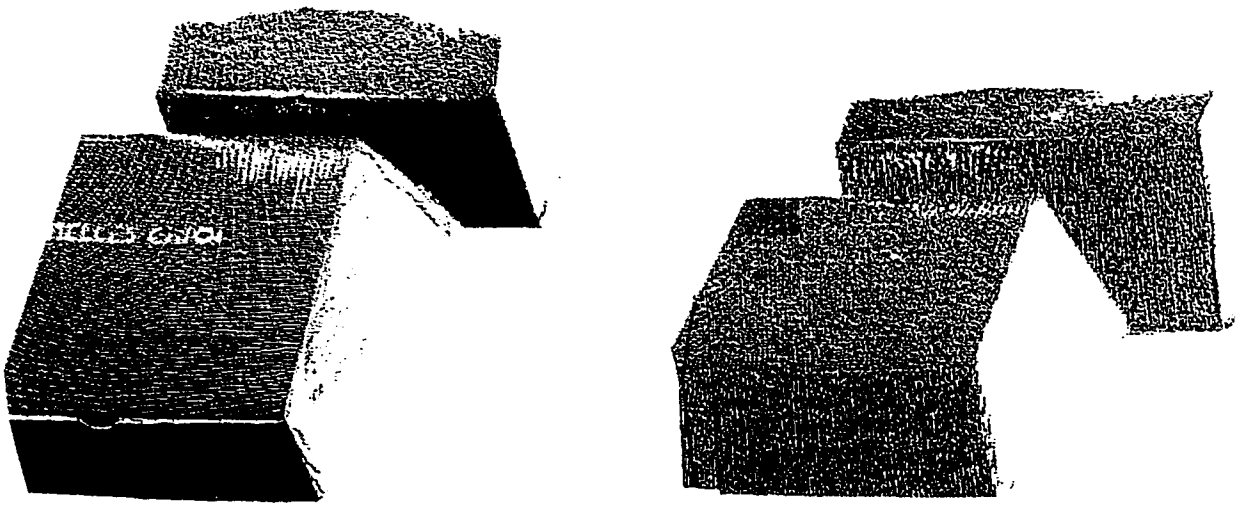
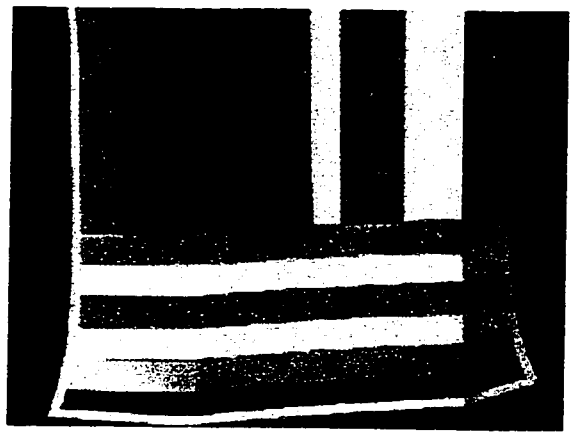


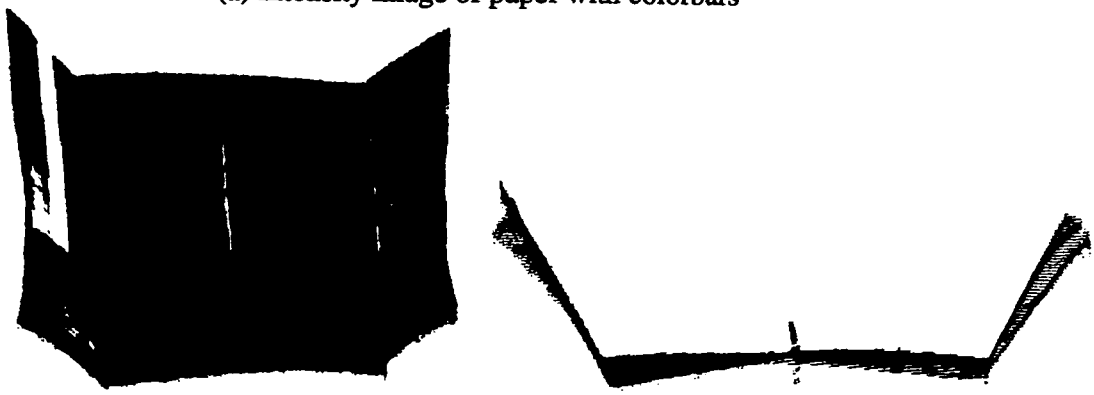
Figure 2.9: Boxes from figure 2.8 reconstructed using traditional (left) and spacetime (right) methods, with data obtained from cameras 0 and 3. Using traditional reconstruction, points on right side of near box must be erroneous since that face was occluded from camera 0. Also, note lack of data on top of box where words occur. On the spacetime reconstruction, observe that no points occur on the occluded face, and depth points are now present where words occur.

is a color image of this object. Figure 2.10(b) is a range image reconstructed using two cameras (cameras 0 and 3) and the spacetime method before we developed the maximum peak test illustrated in Figure 2.7. Note the "hole" in the center of the object. The crease at the junction of the black and white colorbars caused a great deal of inter-reflection, which in turn caused the hole in the image. Notice that where the "hole" occurs there are actually depth points reconstructed, but they are behind the surface of the object. Figure 2.10(c) is the same object reconstructed using the same two cameras and the spacetime method, but with the addition of the maximum peak test. Notice that the hole is gone since there are no erroneous points reconstructed behind the object now. This illustrates the necessity of the maximum peak addition to our algorithm.

Several views ($\sim$16 or so, depending on the object) of the color and range



(a) Intensity image of paper with colorbars



(b) Reconstruction using spacetime method without erroneous peak detection.

(c) Reconstruction using spacetime method and erroneous peak detection.

Figure 2.10: Improvements to inter-reflection problem. Note hole in center of (b), caused by reconstruction of points *behind* object. Hole in center has disappeared from (c) because those points are now being calculated at the proper depth.

data are obtained in this manner. They are then registered and a 3D mesh is constructed for the object in the manner described in the next section.

2.4 Registration and Mesh Generation

Most of the meshes used for the experiments in this thesis are complete reconstructions of 3D objects, with the range data obtained as described in the previous section. This section contains a brief discussion on how the range images are made into a mesh.

2.4.1 Registration

To find a mesh that approximates the entire surface of object, the range maps obtained from the scanning process must be aligned and expressed in a single coordinate system. This process is called registering the range data, and a good deal of research has been devoted to solving this problem. We used the method developed by Pulli [50] for registering the range data.

Automated registration of multiple range images is, in the general case, an unsolved problem. When the separate views are taken under full computer control, such as with a robot or on a computer-controlled turntable, then initial registration parameters are available through the system, though they usually have some associated error. When the views are obtained by moving the object or sensor by hand, this information is unavailable and must be deduced from feature matching, which is unreliable, or estimated by the human operator. Pulli uses a combination of interactive approximation, 2D projection, and distance minimization to register pairs of images, and then uses global optimization techniques to register all pairs together. To obtain an initial registration for a pair of images, the operator selects four point pairs so that each point marks the same surface point in both images. The program finds the

Figure 2.11: Complete registration of several range views of a coffee cup

corresponding 3D points and calculates a transformation matrix to align one view to the other. This initial registration is only approximate, and further refining of the transformation matrix is necessary. The second step renders the second image as if viewed from the location of the first image, and then uses 2D registration techniques to align both 2D images. This improves the transformation matrix further, and then the registration is refined by using a novel combination of 3D techniques. Sets of points that match best are kept for each pair of views. Each view is registered with more than one other view. When all views have been registered pairwise to a few neighboring views, the sets of point pairs are used in a global optimization scheme that obtains the best transformation matrix for each view based on the point pairs. When all the transformation matrices stop changing the best registration of all views has been obtained. This combination of interactive, 2D and 3D methods provides a more accurate, robust registration than the use of 3D methods alone. Figure 2.11 shows the completed registration of a coffee cup.

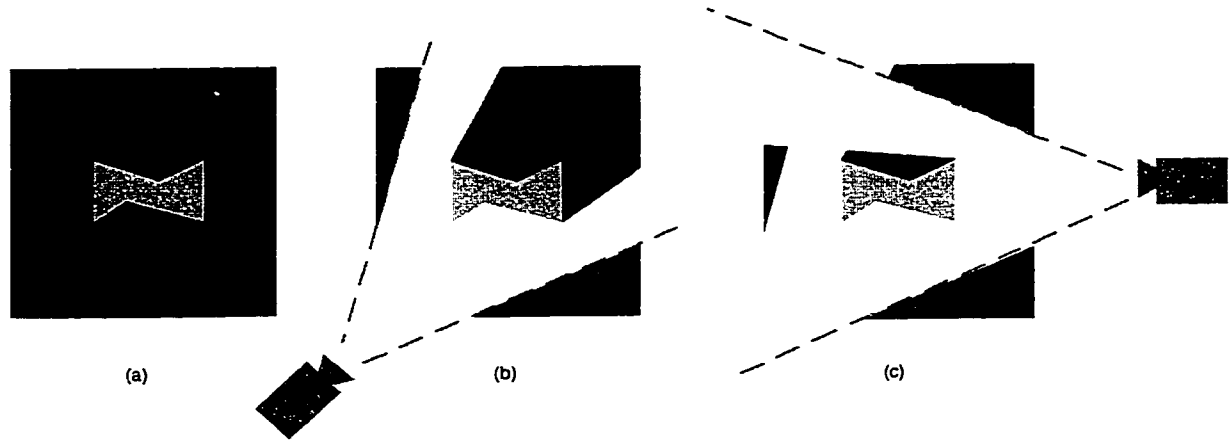


Figure 2.12: The idea of space carving. (a)The working volume contains an object. (b)A scan can prove that some of the volume lies outside the object. (c)More scans remove more space, but the object remains.

2.4.2 Reconstruction

Once all range images are properly registered, the surface must still be extracted. There are, again, several methods available for the task, none of which are perfect. We chose to use Pulli's *space carving* [50] method of surface extraction. The basic idea is to carve away the empty space known to not belong to the object. Figure 2.12 is an illustration of space carving. Figure 2.12(a) shows a working volume with the object to be reconstructed somewhere inside it. In Fig. 2.12(b) there is a scanner with its viewing cone. The volumes of space which lie between the scanner and parts of the object visible to the scanner can be removed since they are known to be empty. If background is visible to the scanner, those volumes between the scanner and the background can be carved away as well. Scanners in different positions, such as the one shown in Fig. 2.12 can carve other parts of space away. With a sufficient number of scanners, we are left only with a connected volume that closely approximates the surface of the object. The Pulli algorithm uses a hierarchical approach and divides the

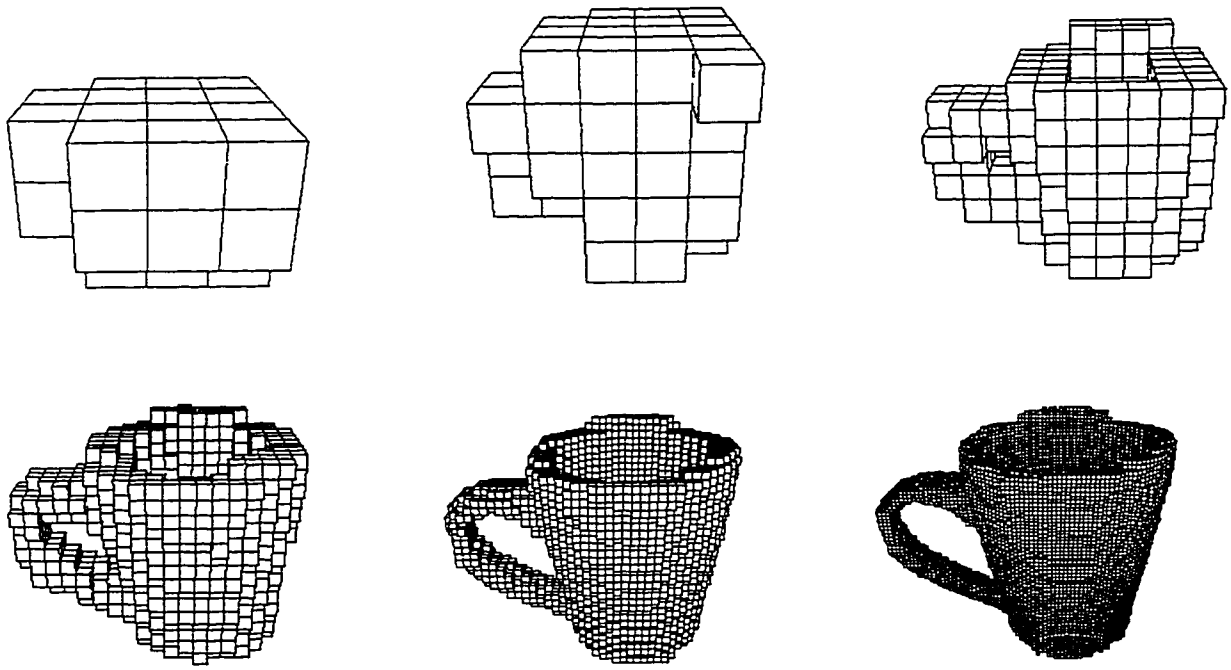


Figure 2.13: Hierarchical carving of cubes.

working volume into cubes. First, the volume is divided into eight cubes and they are all tested to see if they are inside, outside, or on the object. Those that are judged to be outside by any one view are removed from further consideration. Those that are judged to be inside by all views are marked as part of the object. Those that are not inside or outside are further divided into eight cubes, and those cubes are tested. This recursion continues until the desired level of approximation is reached. Figure 2.13 show the progression of the hierarchical space carving method for a coffee cup.

2.4.3 Mesh Generation

Once the desired resolution is reached, the surface representation is extracted by finding the faces of the cubes that form the boundary between "inside" the object and "outside" the object, dividing those faces into triangles, forming ver-

tices where the triangles connect, and extracting the connectivity information. The mesh obtained from the hierarchical space carving method is a good start, but it is very dense and suffers from severe quantization error since all the faces are aligned with the coordinate axes. Smoothing of the mesh and reduction of the number of vertices will provide a better approximation to the actual surface, since very few real objects are made from a collection of cubes. The next step in processing the "cubed" mesh is the application of a specialized low-pass filter developed by Taubin [60] for polygonal meshes. This filter smooths without shrinking, so it removes the spurious noise (in the form of quantization error) from the mesh while still preserving the shape of the object.

Mesh optimization has been studied in great detail in the field of computer graphics. For computer graphics applications the best mesh representation of an object will be one that preserves the shape of the object while using the fewest polygons necessary. Fewer polygons means faster rendering, while more polygons means more surface detail. Since these are competing goals, many mesh optimization algorithms leave fewer polygons in places describing planar-like surfaces, and more polygons where curvature is higher. This allows faster rendering without loss of detail. These algorithms are not generally suited for computer vision applications, since we often want to measure local properties of the data, generally through operations on "neighborhoods." In 2D images and some types of range data, pixels are all a fixed distance apart and the neighborhood of a pixel is just the group of pixels adjacent to it. In a polygonal mesh, we might define the neighborhood of a vertex as all the vertices connected to it. The problem is that the vertices are not a fixed distance apart so neighborhood operations that rely on that characteristic don't work. To control the resolution of a mesh and make it more useful for computer vision work, Johnson and Hebert [38] developed an algorithm that normalizes the distance between vertices as well as minimizing the number of vertices required to represent the

object while preserving its shape. Our dense, smoothed mesh is simplified using Johnson's method which we will now describe.

Johnson and Hebert ascertained five basic requirements for any algorithm designed to control resolution of polygonal meshes for use in computer vision. Those requirements are:

1. Preserve shape
2. Normalize distance between vertices
3. Minimize number of vertices
4. Handle free-form and polyhedral shapes
5. Handle mesh boundaries.

As in computer graphics, items 1 and 3 compete with each other, and we have the added restriction of item 2. Item 4 ensures generalization of the algorithm to most shapes, and item 5 permits the use of meshes generated from a single range image.

The algorithm uses two basic operations iteratively applied to the edges in the mesh to achieve a new mesh of the desired resolution. *Edge-split*, where an edge is broken at its midpoint into two edges, is used to remove edges that are too long. *Edge-collapse*, where an edge is reduced to a point, is used to remove the edges that are too short. The edge-split operation just splits one triangular face into two so it does not change the shape of the mesh. During the edge-collapse operation some shape change is unavoidable since a face is being removed from the mesh. This shape change is kept to a minimum through the use of a *shape change measure*. The edges in the mesh are ordered for edge-collapse based on how much shape change will occur if the edge-collapse

operation is applied. The operation is applied first to the edges that change the shape of the mesh the least. The edge length is also taken into account with a weighting factor, so that of two edges with nearly the same shape change measure, the shortest edge would be collapsed first. One other measure considered when normalizing the mesh is the *accumulated shape change*, which keeps track of how much the shape has changed in the neighborhood of each edge. Global bounds are used to limit the amount of accumulated shape change at each edge. This has the effect of bounding the surface between an inner surface that prevents too much shrinkage and an outer surface that prevents too much expansion.

Figure 2.14 shows the results of the coffee cup cube mesh from Fig. 2.13 after the smoothing operation (Fig.2.14b), simplification to half the original number of faces Fig.2.14c), and simplification to 10% of the original number of faces Fig.2.14d).

2.4.4 Discussion

This method provided us with a reasonable normalized mesh representation of the objects. There is a point, however, where too much simplification causes one (or both) of two problems: either the shape changes so much that it no longer provides a reasonable representation of the object, or the edge lengths must have unacceptable length differences. We expect that the limits of simplification are dependent on both the object and the computer vision processing algorithms being applied to the mesh. In some of our experiments we address the performance of our algorithm using different mesh resolutions.

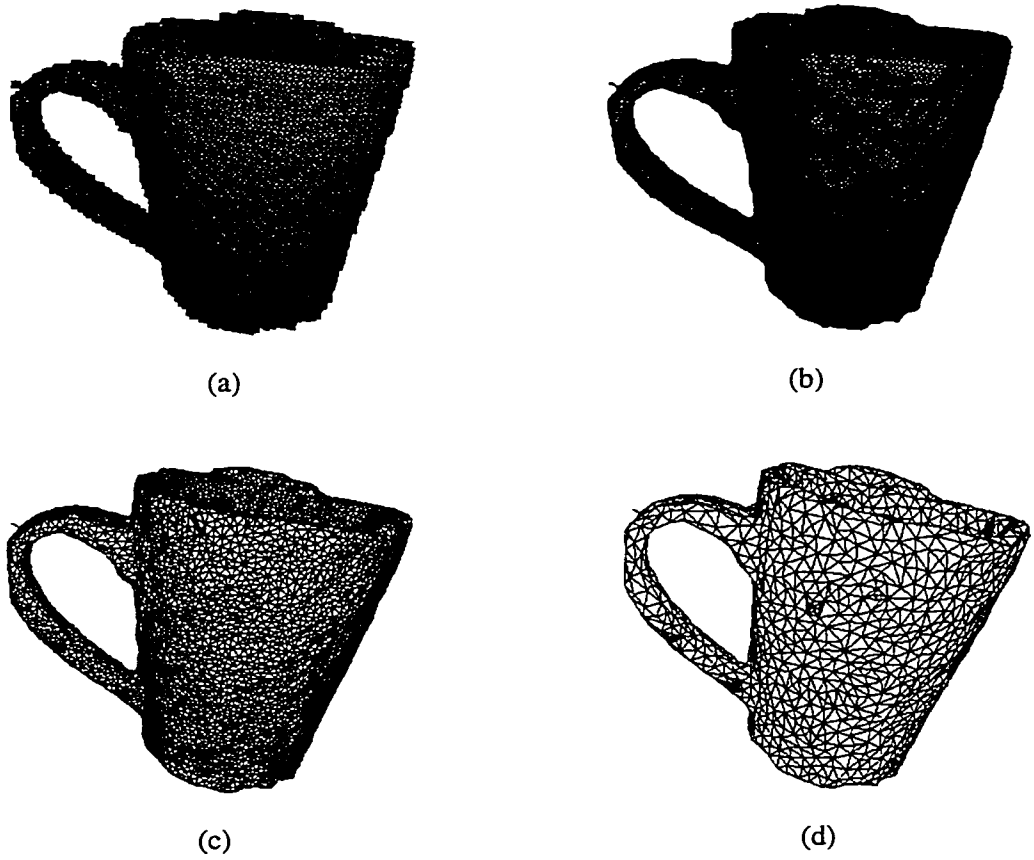


Figure 2.14: Mesh smoothing and normalization. (a) Initial mesh created by hierarchical space carving. (b) Results of smoothing operation. (c) Simplification to half the number of faces. (d) Simplification to 10% of the original number of faces.

Chapter 3

OBJECT MODEL AND MATCHING

3.1 Introduction

Our new methodology uses symbolic models to represent 3-dimensional objects. Symbolic representation of spatial information is not new. In [33], Hernandez presents an entire system based on qualitative spatial knowledge to describe 2D spatial objects and relationships. He proposes a way to extend his system to describe 3D objects based on volumetric primitives. He also provides a comprehensive survey of other work in the area of qualitative spatial representation. The approaches surveyed in this work have one thing in common; they use the qualitative representation for *reasoning* about spatial objects, but do not try to tie the qualitative model to any particular instance of an object. Other researchers have used qualitative spatial information for object recognition. Brooks' [11] ACRONYM system uses spatial relationships of volumetric models based on generalized cylinders to predict the occurrence of 2-D features in gray-tone images. Dickinson *et al.* [15] employ a qualitative shape model based on simple volumetric primitives called geons [9] to obtain the general shape of an object. These primitives are then deformed to fit the range data and obtain a more exact shape. The objects must be composed of the volumetric primitives defined in the system, but our methodology allows more flexibility in that the primitives can be any defined function. Sengupta and Boyer [54] use symbolic information to develop parameterized descriptions of objects to organize a large modelbase for rapid object recognition. In the GRUFF system

[57], Stark and Bowyer use symbolic models to represent categories of objects and use functional characteristics to decide if an object belongs to a particular category. They try to recognize an instance of the object by reasoning about the object based on its ability to function as that category. This system only classifies objects, whereas our system finds features of interest and can provide function-fitting information.

3.2 Abstract Symbolic Model

We have developed an abstract symbolic model to describe any three-dimensional object. This model is useful for reasoning about the object and can be the starting point for domain-specific models to be used in matching. This model is loosely based on the Image Understanding Environment (IUE) [37], which is an ongoing project sponsored by ARPA with the intent to establish a common software environment for image understanding research. Unlike the IUE, however, our model is not dependent on geometry, absolute location, or a geometric coordinate system. Rather than using the geometrically-oriented IUE classes, we adopted IUE concepts for spatial object, part-instance network and topology network and redefined these concepts to fit a non-geometric representation.

This model was motivated by the need to describe and reason about the spatial attributes and relationships of anatomical objects. The abstract model has been adapted to the domain of anatomy and has already demonstrated its usefulness in that domain. The Digital Anatomist Structural Abstraction [45], derived from this model, is an integral part of the Foundational Model of Anatomy [53] which is an ongoing research project in the Department of Biological Structure at the University of Washington Health Sciences Center.

The abstract model consists of a spatial object ontology and three relational networks. The spatial object ontology is a relatively simple hierarchy that de-

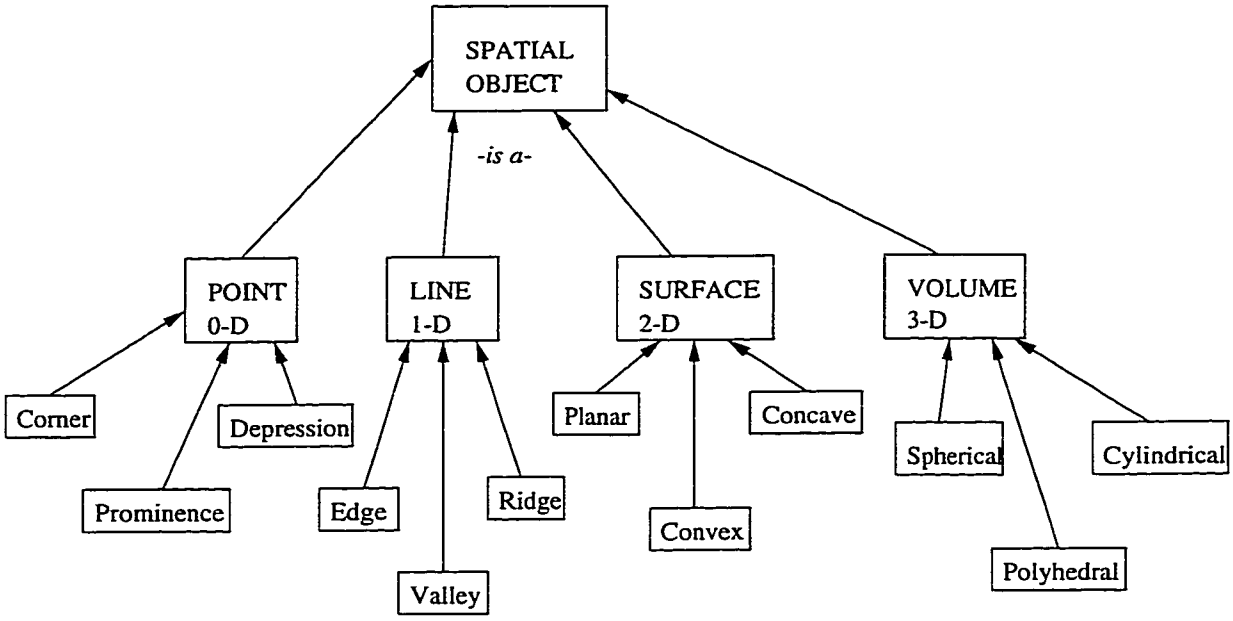


Figure 3.1: The Spatial Ontology.

scribes an object in terms of its dimension and basic shape. The four classes we propose are: point, line, surface, and volume,[1] which represent spatial objects of zero, one, two, and three dimensions, respectively. See Figure 3.1.

The real power of our model lies in the three relational networks. A network consists of links and nodes. The nodes are the spatial objects and the links are the relationships. These networks are influenced by the spatial ontology in that the dimension of the object will determine the kind of relational network the object can participate in.

First is the topology network. The topology of an object describes it in terms of its boundaries, which are themselves spatial objects. In our topology network, the nodes are spatial objects and the links are *-bounded by-* and *-boundary of-*. The nodes must be of a different dimension than the object being

[1]The terms point, line, surface and volume are not used in the strictest geometric sense. We allow some latitude in that we refer to topological as well as geometric information using these terms.

described. Volumes (3-D) are bounded by surfaces (2-D), which are bounded by lines (1-D). Lines (1-D) are bounded by points(0-D), and are a boundaries of a surface (2D). In summary, we can specify the topology network as the pair (O, TH), where O is the object and TH is the topology hierarchy relationship describing that object.

The next relationship is the part-whole relationship. Spatial objects are described in terms of their component parts, which themselves are spatial objects. In the part-of network, the nodes are spatial objects and the links are *part-of* and *has-part*. The nodes must be of the same dimension as the object being modeled. Parts of 3-D objects can themselves only be 3-D and so on. We can specify the part of network as the pair (O, PH), where O is the object and PH describes the has-part/part-of hierarchy relationship.

The third network in our scheme is the Spatial Associations network. It may contain several different relationships, depending on the model. Most networks in the spatial associations network require a coordinate system to fully describe the relationships. If we were using a geometric model, we could define geometric axes, however since our model is symbolic we need to use symbolic axes. The coordinate system is based on the particular object being modeled, and may include such directions as *up-down*, *front-back*, and *right-left*, which correspond to the anatomical directions of *superior-inferior*, *anterior-posterior*, and (roughly) *medial-axial*. The relationship currently established in the spatial associations network is *adjacent-to*.

This is the current design of our abstract object model. Figure 3.2 shows the abstract model of a coffee cup. The cup is the root object so it is not *part-of* anything. It *has-part* **body** and **handle**. The body and handle are both *part-of* the cup, but have no parts themselves. The handle is *bounded-by* the handle surface, but the body is *bounded-by* three surfaces; the inside surface, the outside surface, and the bottom surface. Conversely, these three surfaces

are the *boundary-of* the body, and the handle surface is the *boundary-of* the handle. The inside surface is *bounded-by* the rim, and the outside surface is *bounded-by* the rim and the bottom edge. Since the rim and bottom edge are circular, they have no boundaries.

More complex models are represented by Figures 3.3 and 3.4 which show the partial abstract models of an airplane and lung .

3.3 Working Model

The abstract symbolic model described in the previous section is useful for reasoning about the object, as discussed in [53], but it has no mechanism to identify itself with a real object. The *working model* provides that mechanism. The working model is designed to model only *part* of the abstract model dependent on the type of geometric function used in the algorithm. If volumetric functions are desired, the working model is based on 3D functions and will therefore model the volumes of the abstract model. If surfaces are of interest and surface fitting functions can be obtained, the working model will model the 2D parts of the abstract model. In the implementation chapter, we will show how we use the working model to model the 1D aspects of the abstract model.

The working model consists of the set of features F of the object we desire to find and a set of global constraints R associated with those features. Each feature has five main properties: *label*, *working-set-types*, *shape-type*, *shape-parms* and *detectability*. The label is just the name of the feature. Working-set-types and shape-type are properties used by the algorithm that detects the feature. There may be any number of working sets associated with a feature and for each there will be a working-set-type. The working-set-type limits the sets of mesh points that may be associated with the feature. Shape-type limits the shape of the detected feature and has parameters, a fit-function, and an

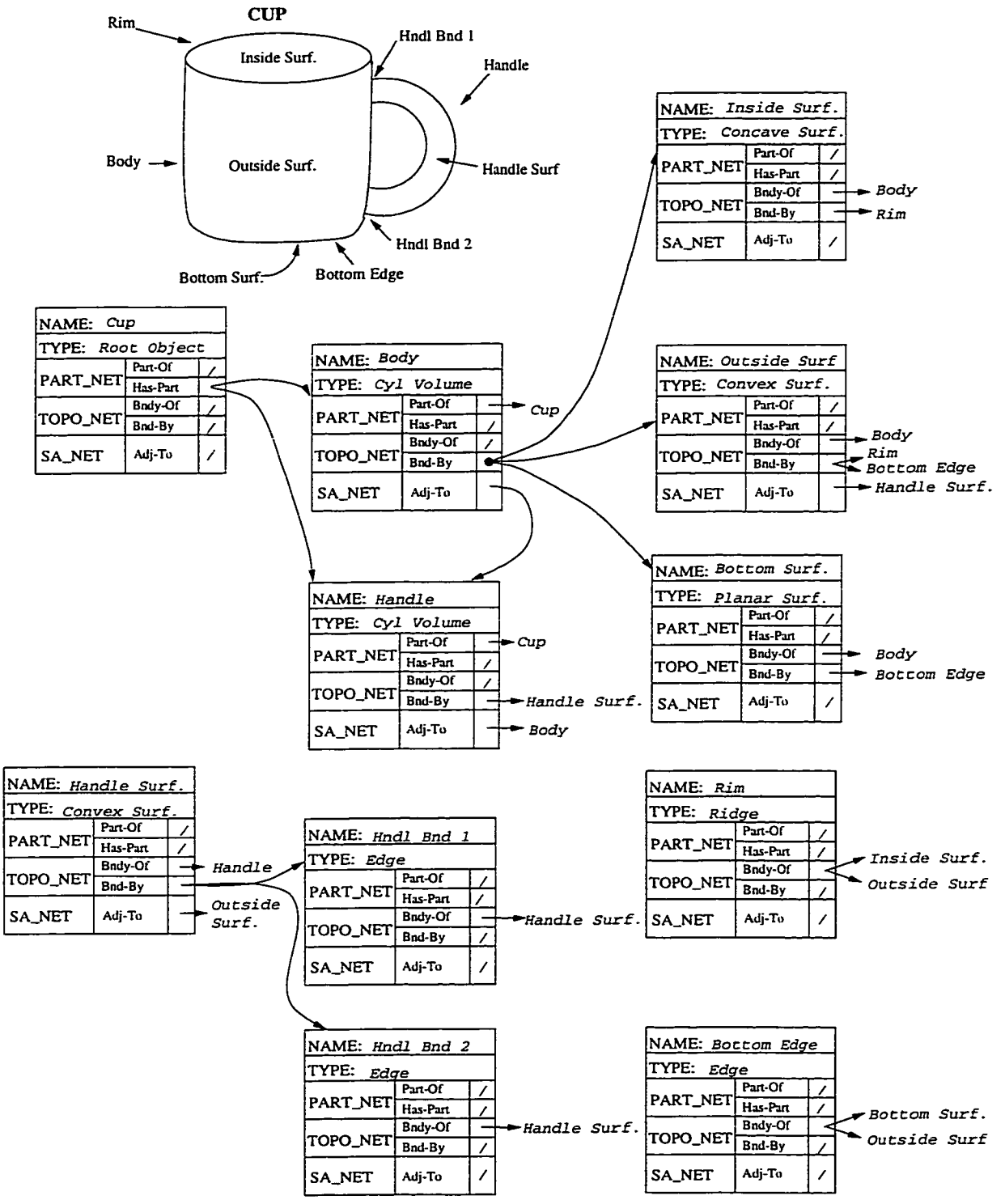


Figure 3.2: Abstract Model of Coffee Cup.

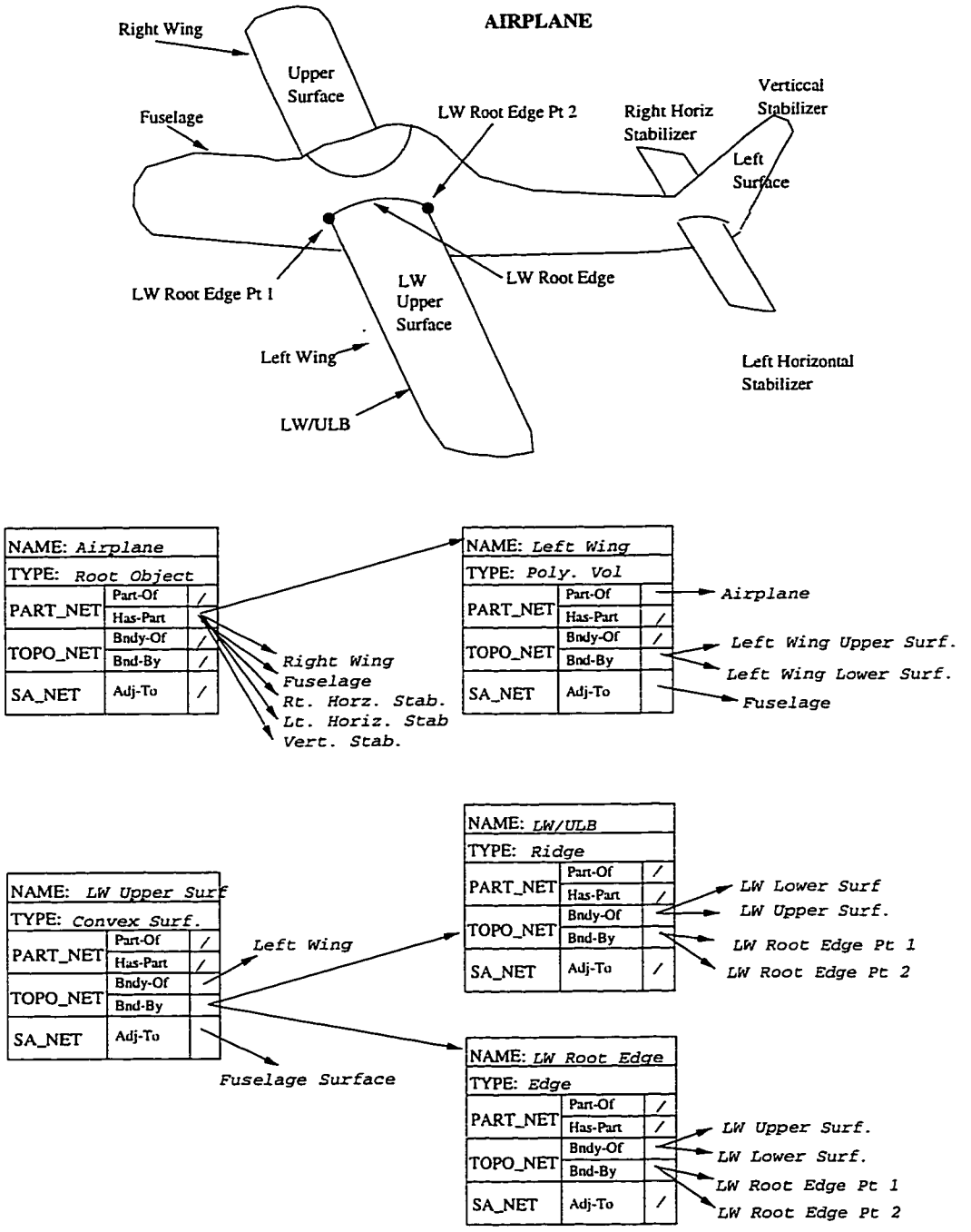


Figure 3.3: Abstract Model of Airplane. Due to complexity, only a partial model is shown here.

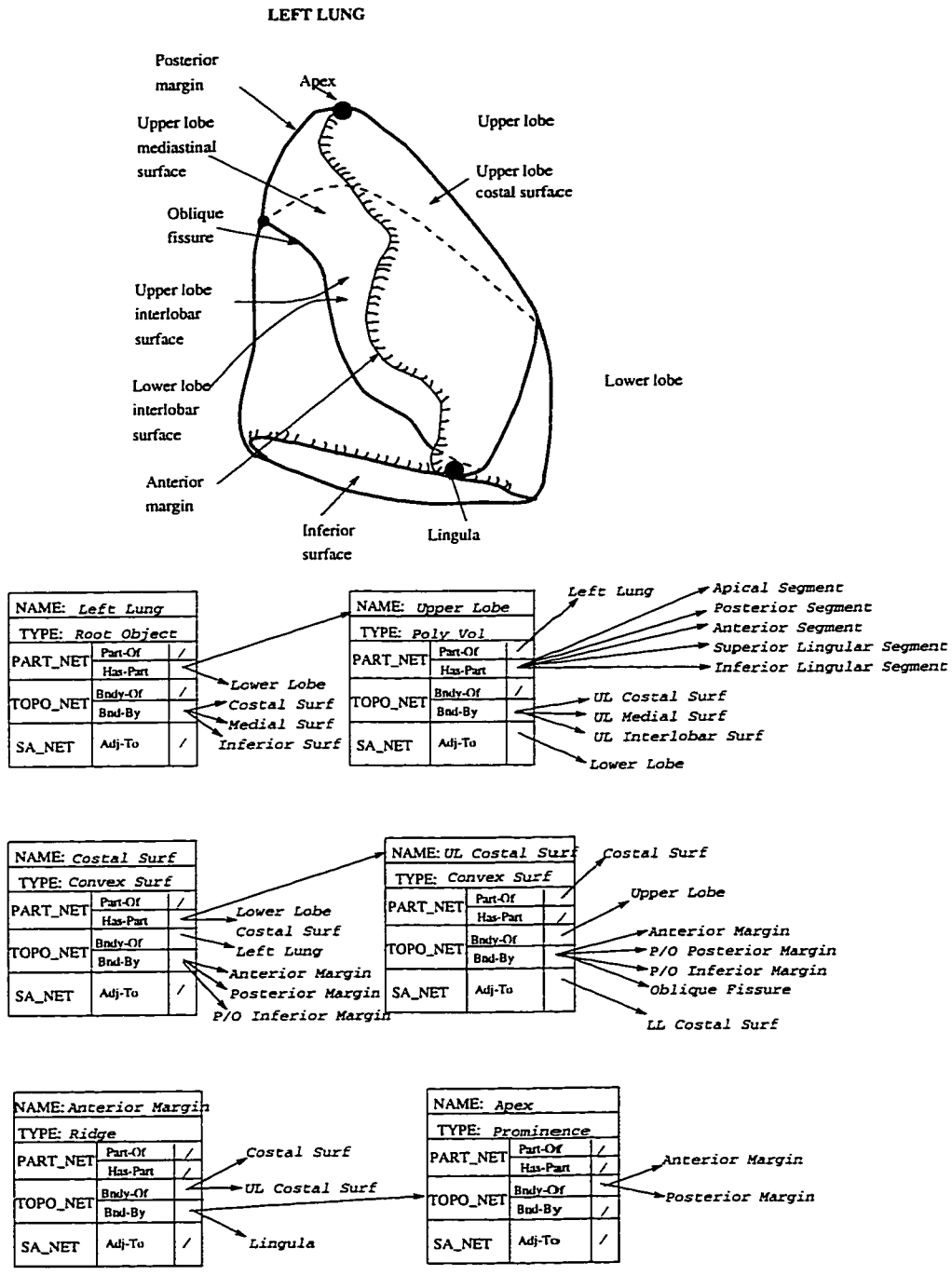


Figure 3.4: Abstract Model of Left Lung. Due to complexity, only a partial model is shown here.

error function associated with it. Shape-parms is the set of parameter values for that feature. Detectability is the likelihood of finding this particular feature; it is used in the feature labeling algorithm.

3.4 Matching

Given a working model of a 3D object class and a 3D mesh data set representing a sensed instance of that class, *matching* is the process of finding the features of the model as subsets of the mesh points that satisfy the shape requirements and relationship requirements of the model. Our proposed matching methodology has three stages: working set construction, feature construction, and feature labeling.

3.4.1 Definitions

The first stage of the matching methodology is bottom-up and its purpose is to find, for each feature of the working model, one or more candidate sets of points in the mesh where this feature may be found. The candidate sets of points are called *working sets*. More formally, a working set $W(f)$ of mesh M for feature f with working-set-type t is a subset $W(f) \subseteq P(M)$ of the point set $P(M)$ of the mesh M satisfying:

1. $W(f)$ is a connected point set.

2. t is a valid type for feature f.

3. $W(f)$ satisfies the requirements for type t.

Given a working model Q with feature set F, the first stage of matching is to identify for each feature $f \in F$ a set of working sets $S(f)$ for that feature.

Thus the output of this stage is

$$S(F) = \{S(f) | f \in F\}$$

The second stage is top-down, and its purpose is to further reduce the number of candidate sets of mesh points for each feature by ensuring the working sets still under consideration have the proper general shape and the parameters of that shape fall within the range of the shape parameters of that feature. These remaining working sets, in conjunction with the fitted shape-type of the feature, are called *constructed features*. In other words, a constructed feature $C(f)$ of mesh M for feature f with shape-type s and shape-parms m is a 3-tuple $C(f) = (W(f), G(W), E(W))$, where $W(f) \subseteq S(f)$, $G(W)$ is the set of parameters p resulting from a best-fit of s to $W(f)$ and $E(W)$ is a fit-error metric associated with $G(W)$. $C(f)$ must satisfy the following conditions:

1. s is a valid shape-type for feature f.

2. Each parameter $p \in G(W)$ falls within the given range of the corresponding parameter m in feature f.

3. $E(W)$ must not exceed some given error threshold t_{err}.

Note that a constructed feature can be formed from more than one working set. This will allow generation of features that may be more robust than those generated from just one working set. For example, the fillets where the handle on a coffee cup connects to the body are usually detectable, but they are often small and can be mistaken for noise in the mesh. We use a constructed feature that consists of a line segment fit to *both* fillets, then base the constraints on properties of that line segment. See Figure 3.5.

Given a working model Q with feature set F, the second stage of matching is to construct for each feature $f \in F$ a set of constructed features $T(f)$ for that

feature. Therefore the output of the second stage is

$$T(F) = \{T(f) | f \in F\}$$

The last stage of the matching methodology is also top-down, and its purpose is to find a consistent labeling from the working model features to the constructed features. Let $f \in F$ be a working model feature and $T(f)$ be its set of constructed features. Let R be the set of constrained relationships in the working model; $R = \{r_1, \ldots, r_m\}$, where each $r_i \in F^n \times \Upsilon$ is a labeled tuple of features $(f_1, \ldots, f_n, \rho)$ in which $f_i \in F$, and $\rho \in \Upsilon$ is the type of relationship. A *consistent labeling* $h : F \rightarrow T(F)$ is a function that maps model features to constructed features and must satisfy the following:

1. h is one-to-one, i.e., $h(f_i) = h(f_j) \Rightarrow f_i = f_j$

2. If a relationship between features is in R, then the parameters in the associated constructed feature must also satisfy the relationship.

$$\rho(f_i, \ldots, f_j) \Leftrightarrow \rho(C(f_i), \ldots, C(f_j))$$

3.4.2 Algorithm

This section provides details of the algorithms used to accomplish the stages described in the above methodology. We also define the function p as follows. Let F be the set of N features in the working model. For each feature, $f_i \in F$ $i = 1, \ldots, N$, let function $p(f_i)$ return a *property vector* containing all properties of feature f_i.

Working Sets

As defined previously, the mesh M contains a set of points, $P = \{p_1, \ldots, p_n\}$. From this set of points, we select a set W of *working point sets*. Each working

point set $w_i \in W$ is a subset of P. Any point p_k may only be part of one w_i. Determination of the $w_i \in W$ can be based on a number of criteria, mainly depending on the desired representation of the feature properties in the model.

Constructed Features

The next part of the algorithm consists of three main steps for each feature f: 1) select the set of candidate working sets $S(f)$ for that feature 2) fit the shape type of the feature to each $W(f) \in S(f)$, and 3) form the set of *constructed features* $T(f)$. The selection algorithm is based on the feature property *working-set-type*, which is one of the types defined in the working model. The selection algorithm is as follows:

for each $f_i \in F$ do
  current_properties=$p(f_i)$
  for each working-set-type $w_j \in$ current_properties.working-set-types
    w_j_$candidate_stack_{f_i}$=NIL;
    for each $w_k \in W$
      if $w_k.type == w_j.type$
        push(w_j_$candidate_stack_{f_i}$, w_k)

The fitting algorithm is based on the feature property *shape-type*. It fits the shape function for each feature to each $W(f)$ for that feature. If the fitted shape parameters fall within the parameter range for f_i, a *constructed feature* $C(f_i)$ is formed. In this manner, a set of constructed features $T(f_i)$ is generated for each f_i. The fitting algorithm is:

for each $f_i \in F$ do
  current_properties= $p(f_i)$

Wc = the set of all working set groups, where each group contains one working set from each $w_j\_candidate\_stack_{f_i}$

for each $wc_j \in Wc$

w_shape_parms=fitShape(current_properties.shape,wc_j,fit_err)

if w_shape_parms ∈current_properties.shape_parms

form_constructed_feature(wc_j,w_shape_parms,fit_err)

Once the set of constructed features $T(f_i)$ is formed for each f_i it is passed to the feature labeling stage.

Labeled Features

The feature labeling stage of the algorithm uses the constructed features as potential correct labelings, and the global constraints in the model to resolve the multiple labels and features into the one correct consistent labeling.

Consistent Labeling The *consistent labeling problem* (CLP) can be defined as follows:

Given a set of M units $U = 1, .., M$ and a set of N labels $L = 1, .., N$, find a mapping $U \mapsto L$ such that the set of given constraints C over the relationships among the unit-label pairs is satisfied [32].

The consistent labeling problem may have one solution, many solutions or no solution at all. A classic example of the consistent labeling problem is the *N-queens* problem. An $N \times N$ chessboard and N queens are given. The task is to place the N queens on the chessboard in such a manner that no queen can capture any other. We can think of the units as the rows and the labels as the columns. The constraints are that no queen can be in the same row, same column, or on the same diagonal as any other. A consistent labeling solves the

problem by telling us, for a given row, in which column to place a queen so that the constraints are satisfied. Note that for $N < 4$ this problem has no solutions, but for $N \geq 4$ it has more than one solution.

Consistent labeling problems generally fall into three categories:

1. All constraints must be satisfied

2. The number of constraints not satisfied (or the error) must be less than some threshold

3. The error must be minimized. The minimal error mapping is the solution. If the error is greater than a threshold, though, we may discard it and say there is no solution as in 2.

The method used to solve the problem is dependent on the nature of the problem and the desired results. The following is a summary of the common methods used to solve these problems; see Haralick and Shapiro [31] for a more detailed discussion.

A typical method for solving the first type of consistent labeling problem is the *backtracking tree search*, which begins by assigning any label to the first unit of U. This is a node at the first level of the tree. The algorithm then selects the second unit of U and constructs the children of the first node, which are nodes that map the second unit of U to all labels. Constraints are checked as each node is constructed so any node that violates the constraints can be removed immediately. This process continues until the tree has as many levels as there are units. Any paths from the root to the deepest possible level are consistent labelings. On its own this algorithm is very time consuming and inefficient, with exponential time complexity. Improvements can be made using techniques such as *discrete relaxation* to prune the search tree. Discrete relaxation basically checks for incompatibilities between a future unit-label pair

and other future unit-label pairs. For example, if for a particular unit u with label l, there is no label l' for some other unit u', then unit-label pair (u, l) need never be considered. This technique can be performed before starting the tree search and, for very tightly constrained problems, can result in great reduction or even elimination of the tree search. For problems with looser constraints the algorithm should be applied at each node of the tree, which will further constrain the search.

Although many consistent labeling problems (such as the *N-queens* problem) fall into the first category, most computer vision consistent labeling problems do not. We are usually trying to match labels from a perfect model to data extracted from real, imperfect images. To expect all constraints to be met is overly optimistic and almost always results in failure to find a solution. The other two categories of CLPs allow for some error and are suitable formulations for computer vision problems.

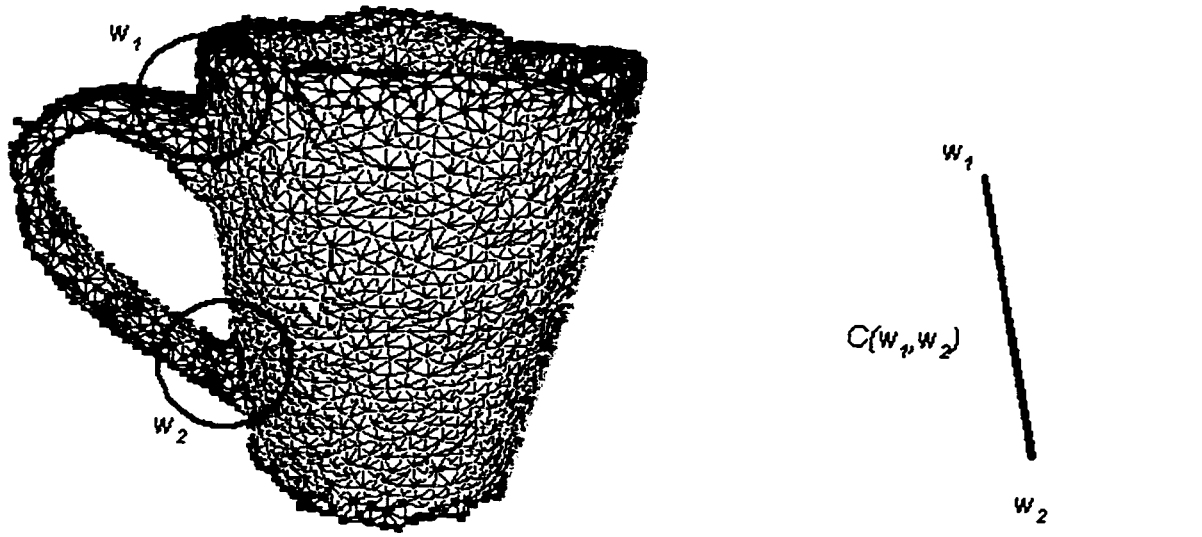
Continuous relaxation is one method of allowing some error in CLP solutions. Instead of removing possible labels from the label set L_i of a unit i as discrete relaxation does, continuous relaxation updates the probability of a unit u having a given label l after each iteration. Initial probabilities are defined by a set of functions giving *a priori* probabilities. After each iteration of labeling, a new probability is computed for each unit-label pair. This probability is based on the previous probability and information on compatibility with other unit-label pairs. Iterations can continue until a satisfactory mapping (if one exists) is found.

Optimization is another method of solving these types of consistent labeling problems. An error function is defined for the mapping based on the unit-label constraints. Since some constraints may be more important than others, this error function can allow a higher penalty for violating some constraints over others. Once the error function is established, optimization techniques are

used to minimize the error function by adjusting the unit-label pairs.

Constraint-based Tree Search In our system, the units are the features from the working model and the labels are the constructed features from the data. Each feature has the potential of being assigned one of several constructed features. Our consistent labeling problem is to assign a single constructed feature to each model feature in such a manner that the constraints between the model features are satisfied by the constructed feature. A constructed feature may come from one or more working sets. Each working set can produce several different constructed features. Only one constructed feature per working set may become the label of one of the working model features. Figure 3.5 is an example of a constructed feature produced from two working sets. The line segment fit to the two working sets w_1 and w_2 is the feature we are trying to match. Our approach to this problem is to use a *constraint-based* tree search which attempts to find the labeling based on likelihood of finding features to satisfy the constraints. Arman and Aggarwal's strategy trees [1] are similar in philosophy in that they also attempt to reduce the search space by looking for the most detectable features. That system was very different from ours since it was a CAD-based recognition system and therefore relied on the availability of a CAD model for the object to be recognized. Our method is top-down in that we examine the constraints first to find the features we want to search for initially. We describe our approach assuming pairwise constraints for simplicity, though the method is not restricted to only pairs.

The set of binary global constraint relationships is $R = \{r_1, r_2, .., r_m\}$, $r_i \in F^2 \times \Upsilon$. Let $r_i = (f_j, f_k, \rho)$ where f_j and f_k are model features and ρ is the relationship type. We first calculate the *detectability* of r_i, $det(r_i) = det(f_j) * det(f_k)$ and rank-order the r_is based on detectability. Starting with the highest ranked r_i, and assuming feature f_j in r_i has maximum detectability (of all features in



(a) Coffee cup with working sets of interest shown in cyan

(b) Handle feature $C(w_1,w_2)$ constructed from two working sets.

Figure 3.5: Constructed feature formed from two working sets.

r_i), get $T(f_j)$, which is the set of constructed features for f_j. These will be the nodes of the first level of the tree. Get $T(f_k)$ and for each $C(f_k) \notin C(f_j))$, check for satisfaction of ρ between constructed features $C(f_j)$ and $C(f_k)$. Nodes of the tree that have no leaves because there are no constructed features that satisfy relationship ρ are removed from further consideration. The tree is traversed in this manner until all relationships are satisfied.

Most relationships between the features will not be of the satisfied/not satisfied type. Instead, a relationship will be satisfied if it meets the relationship criteria within some error ϵ. For example, if two features are constrained to be parallel to one another, it is not likely the constructed features will be exactly parallel. We say they satisfy the relationship if they are parallel within ϵ. The result of the tree search described above may be more than one consistent labeling. If this is the case, the labeling that best satisfies the constraints is chosen as the correct one.

Table 3.1: Working sets from Figure 3.6b that satisfy relationships ρ_1 (parallel), ρ_2 (perpendicular), and ρ_3 (centroid-distance).

	Relationship		
	ρ_1	ρ_2	ρ_3
Satisfied	w_1, w_3	w_1, w_2	w_2, w_3
By	w_1, w_4	w_2, w_3	w_3, w_4
Pairs	w_3, w_4		

The tree search described above is best illustrated with a simple example. The shape types and relationships will be formally defined in the next chapter, but we will use some of them here for illustration purposes. We will use one working set type wt and one shape type st, which is a circle. The relationships are ρ_1(parallel), ρ_2 (perpendicular), and ρ_3 (centroid distance). Figure 3.6a shows an object with three circular features, f_1, f_2, and f_3. Figure 3.6b shows four working sets $w_1, \ldots, w_4$ with the shape type st fit to each of them. The features are in the format f_i=*(label, working set type, shape type, shape parms, detctability)* and are defined by:

$f_1 = (l_1, wt, st, sp_1, 1)$

$f_2 = (l_2, wt, st, sp_2, 0.75)$

$f_3 = (l_3, wt, st, sp_3, 0.9)$

The relationships $R = \{r_1, r_2, r_3\}$ are defined:

$r_1 = (f_1, f_3, \rho_1)$

$r_2 = (f_1, f_2, \rho_2)$

$r_3 = (f_2, f_3, \rho_3)$

Table 3.1 shows the working sets that can satisfy the three relationships.

Since relationship ρ_1 has the highest detectability ($det(f_1) * det(f_3) = 0.9$)

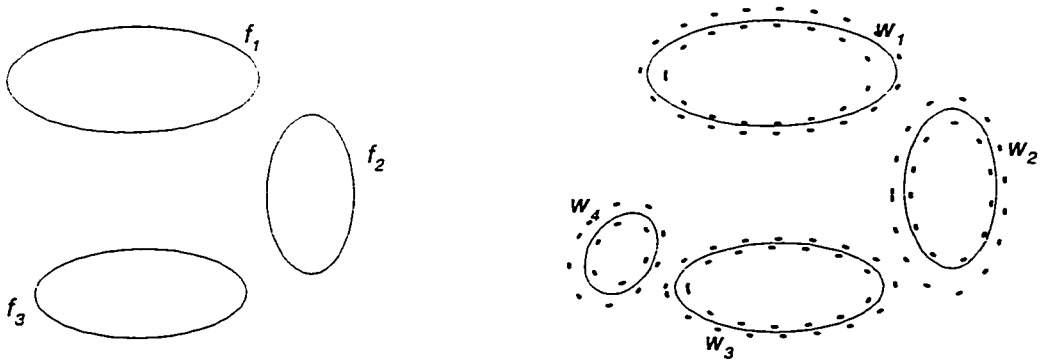


Figure 3.6: Features and shape-fitted working sets for tree search example. (a) Object contains three circular features. (b) Circles fit to working sets. Extra working set w_4 is caused by noise or other imperfections in original mesh.

the constructed features associated with f_1 will be the nodes of the first level of the search tree. Figure 3.7 illustrates the pruning of the tree as the search progresses. Solid lines indicate potential labelings while the dashed and dotted lines indicate satisfied and unsatisfied constraints. Figure 3.7a shows the first two levels of the tree before any constraints are tested. Figure 3.7b shows the tree after ρ_1 is tested. Since working set w_2 does not satisfy constraint ρ_1 with any working set, it is removed from consideration as a possible labeling for f_1 and f_3. It will be reconsidered for labeling further down in the tree. Figure 3.7c shows the tree after ρ_2 is tested, and 3.7c after ρ_3 is checked. Extra working set w_4 is eliminated in this stage because it does not satisfy constraint ρ_3 given the remaining potential labelings. Figure 3.7d shows the final consistent labeling.

It may not be possible to generate constraints for all features we desire to label. There are two different methods to manage this situation. One way is to label all features that have constraints then label the the remaining features based on how well the required shape function fits the working set. Another

method assumes the working model features are associated with the abstract model features. All features that have constraints should be labeled, then relationships found in the abstract model employed to reduce or eliminate incorrect labelings.

3.5 Contributions

The major contributions in this chapter are:

- Development of a new way of modeling 3-D objects. The new method uses an abstract symbolic model coupled with a working model. The working model contains quantitative parameters that can be compared to actual values obtained by fitting specified functions to the range data.

- Development of a two step model-directed search procedure that takes advantage of both bottom-up and top-down techniques.

- Introduction of *constructed features*, which have a structural foundation in the data but can exploit local relationships between groups of points having similar characteristics.

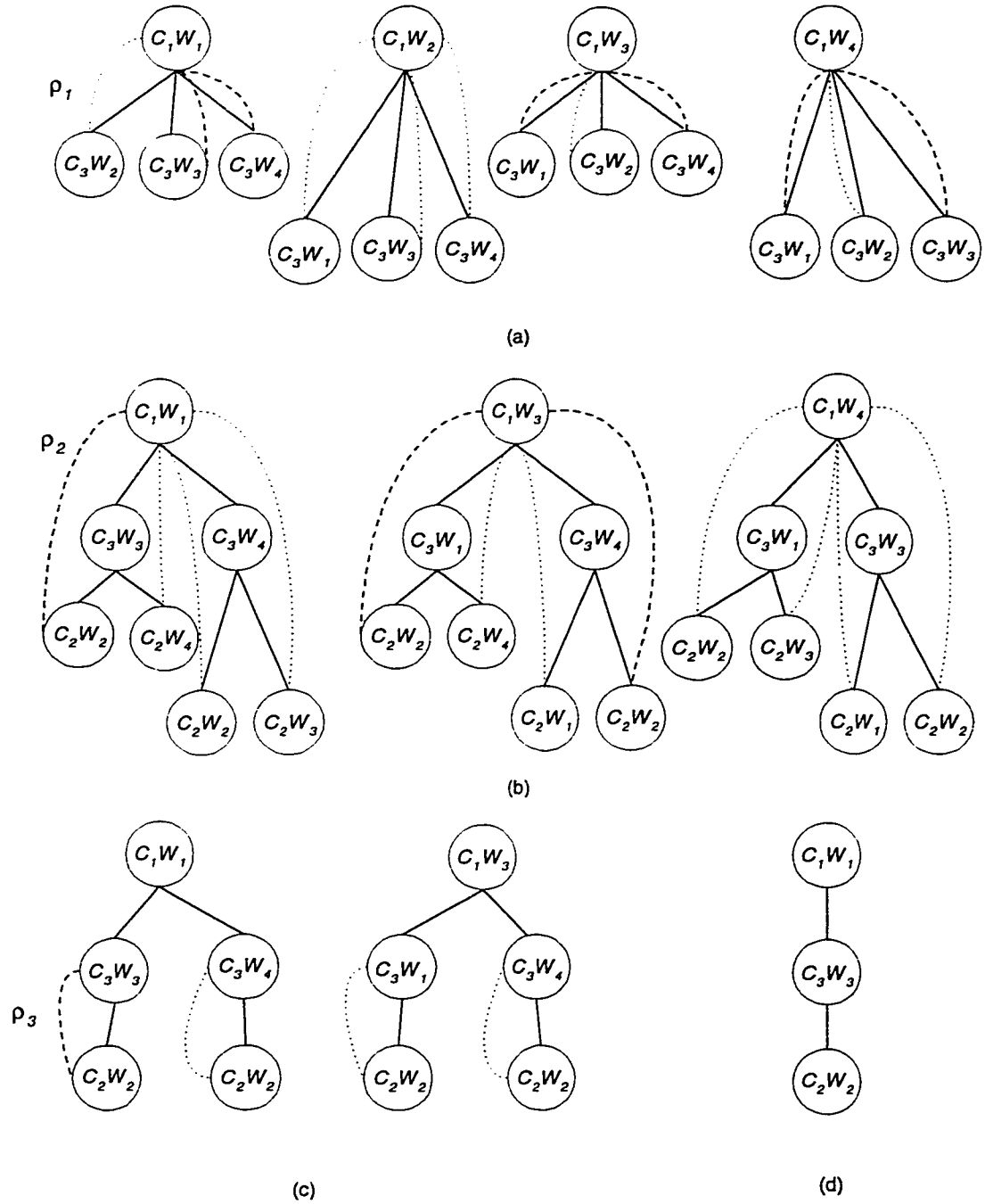


Figure 3.7: Example of constrained tree search using the working sets and features from Figure 3.6. Solid lines are potential labelings and dotted and dashed lines are constraint paths. Dashed lines indicate that a constraint is satisfied, while dotted lines indicate and unsatisfied constraint. (a) Before constraints are tested. (b) After ρ_1 tested. (c) After ρ_2 tested. (d) Final labeling after ρ_3 tested.

Chapter 4

SYSTEM IMPLEMENTATION

4.1 Introduction

This chapter describes the implementation of a prototype system using the model described in the previous chapter. We developed our working model on the premise that we wanted to find the one-dimensional features described in the abstract model of the object. We believe that if the *separators* (edges, or other places where two surfaces come together) in an object can be located, we can then use the abstract model for segmentation of the object into surfaces and volumes.

We developed four working set types based on curvature of the surface in a neighborhood. We implemented three functions for shape type, and allowed five types of constraint relationship between the features. This chapter first reviews curvature and describes our method of finding it. We then discuss the functions used for shape type and our method of fitting those functions. Finally, we examine the relationships used in our implementation.

4.2 Surface Curvature Computation

Since we are looking for separators in the object mesh, curvature was a natural choice as a basis for working-set-types. Wherever two surfaces join a change in curvature occurs, or there would be no basis for labeling them as two different surfaces. Determining curvature in range data is a well-understood and thoroughly researched problem. Several methods have been developed for es-

timating curvature; in [26] Flynn and Jain compare five different curvature estimation methods and discuss the performance of each.

Most curvature estimates are done using a functional surface representation where the surface can be specified as the height $f(u,v)$ above a support plane defined by the two coordinates (u,v). This type of surface is called a *Monge patch*, and a 3D point on the surface is given by $X(u,v) = (u,v,f(u,v))$. For any point p on such a surface, we can choose two orthogonal vectors in the tangent plane and observe the behavior of the surface in those directions. Each direction specifies a curve in X; its curvature measures its tendency to 'bend' out of its tangent field. There are two parts to the curvature: a component due to the way the curve is imbedded in the surface, and a component due to the way the curve changes in 3D. It is this last part, also called the *normal curvature* that we are interested in. The two orthogonal vectors in the tangent plane to the surface at p can be chosen in such a manner that the normal curvatures they yield are the maximum and minimum possible over all such choices. These values are known as the principal curvatures of the surface at p, and are represented as κ_{max} and κ_{min}. Gaussian curvature, $K = \kappa_{min} * \kappa_{max}$, and mean curvature, $H = (\kappa_{min} + \kappa_{max})/2$, are often used by computer vision researchers instead of principal curvature due to their useful invariant properties. The H and K curvatures can be used for classifying surfaces into general shape types, such as saddle, ridge, valley, etc., based on the signs of H and K. For a Monge patch surface, the Gaussian and mean curvatures can be calculated by

$$K = \frac{f_{uu}f_{vv} - f_{uv}^2}{(1 + f_u^2 + f_v^2)^2}$$

$$H = \frac{f_{uu} + f_{vv} + f_{uu}f_v^2 + f_{vv}f_u^2 - 2f_uf_vf_{uv}}{2(1 + f_u^2 + f_v^2)^{\frac{3}{2}}}$$

where f_u, f_v, f_{uu}, f_{uv}, and f_{vv} are the partial derivatives of f. If a function is fit to a local neighborhood of range points, these partial derivatives are easily

obtained and H and K can be calculated. If κ_{min} and κ_{max} are desired they are just the roots of the equation [7]

$$\kappa^2 - 2H\kappa + K = 0$$

or

$$\kappa_{1,2} = H \pm \sqrt{H^2 - K}$$

Curvature estimation is generally done by fitting small surface patches to neighborhoods and calculating the partial derivatives of those functions. Our meshes, since they are based on not a single range image but several range images merged, are not Monge patches. The techniques above are not easily applied to our data. We could create an artificial support surface in the neighborhood of each point, but we are interested in determining amount of curvature not local surface shape, and there are simpler ways to accomplish this.

Instead of fitting surface patches, we chose a numerical method, used by Itter and Jain [36], which produces curvature estimates at a point p by examining the orientation change between it and its neighbors. No support surface or analytical function is needed, but an estimate of the surface normal at each point is required. There are many ways to obtain a surface normal estimate. We used a common, straightforward method of fitting a least squares plane to the points in the immediate neighborhood of p and using the normal to the fitted plane, $\mathbf{n}_p$. Once the normals are calculated, we estimate the curvature from p to each neighboring point q by

$$\kappa_{p,q} = \begin{cases} \frac{|\mathbf{n}_p - \mathbf{n}_q|}{|p-q|}, \text{ if } \mid p - q \mid \leq \mid (\mathbf{n}_p + p) - (\mathbf{n}_q + q) \mid \\ -\frac{|\mathbf{n}_p - \mathbf{n}_q|}{|p-q|}, \text{ otherwise} \end{cases}$$

The curvature is estimated in this manner for a point p and all of the points q in the neighborhood of p. The maximum and minimum of these values are

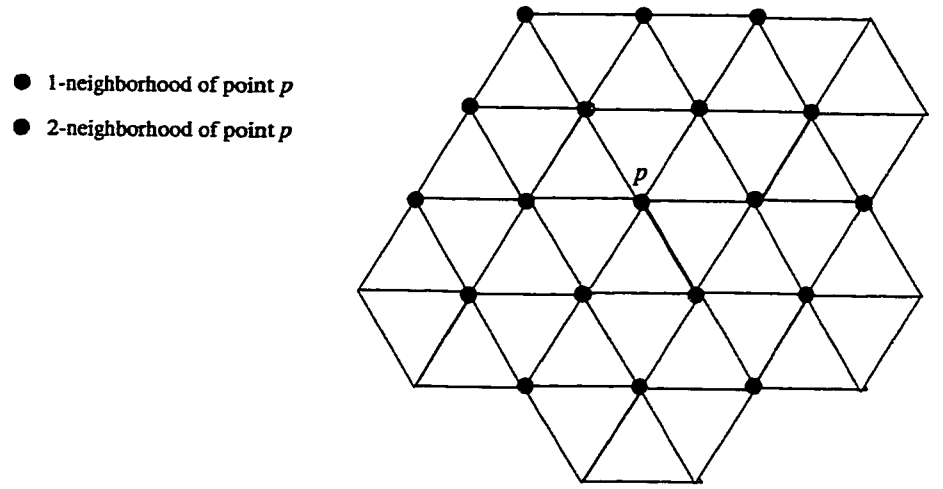


Figure 4.1: Neighborhoods defined on a partial mesh

taken as estimates of the principal curvatures of p. This method was originally used for a Monge patch, but since it does not depend on a support surface we can adapt it for our use.

4.3 Working Set Types

In our prototype system we defined four working set types based on local curvature. They are PLANAR (very low curvature), CONCAVE (high magnitude minimum curvature, CONVEX (high magnitude maximum curvature), and OTHER (curvature between PLANAR and CONCAVE/CONVEX). We used the method described above to determine curvature, with a neighborhood N of vertex p defined as all the vertices N edges away from p and connected to p. Figure 4.1 illustrates the 1- and 2-neighborhood of a partial mesh.

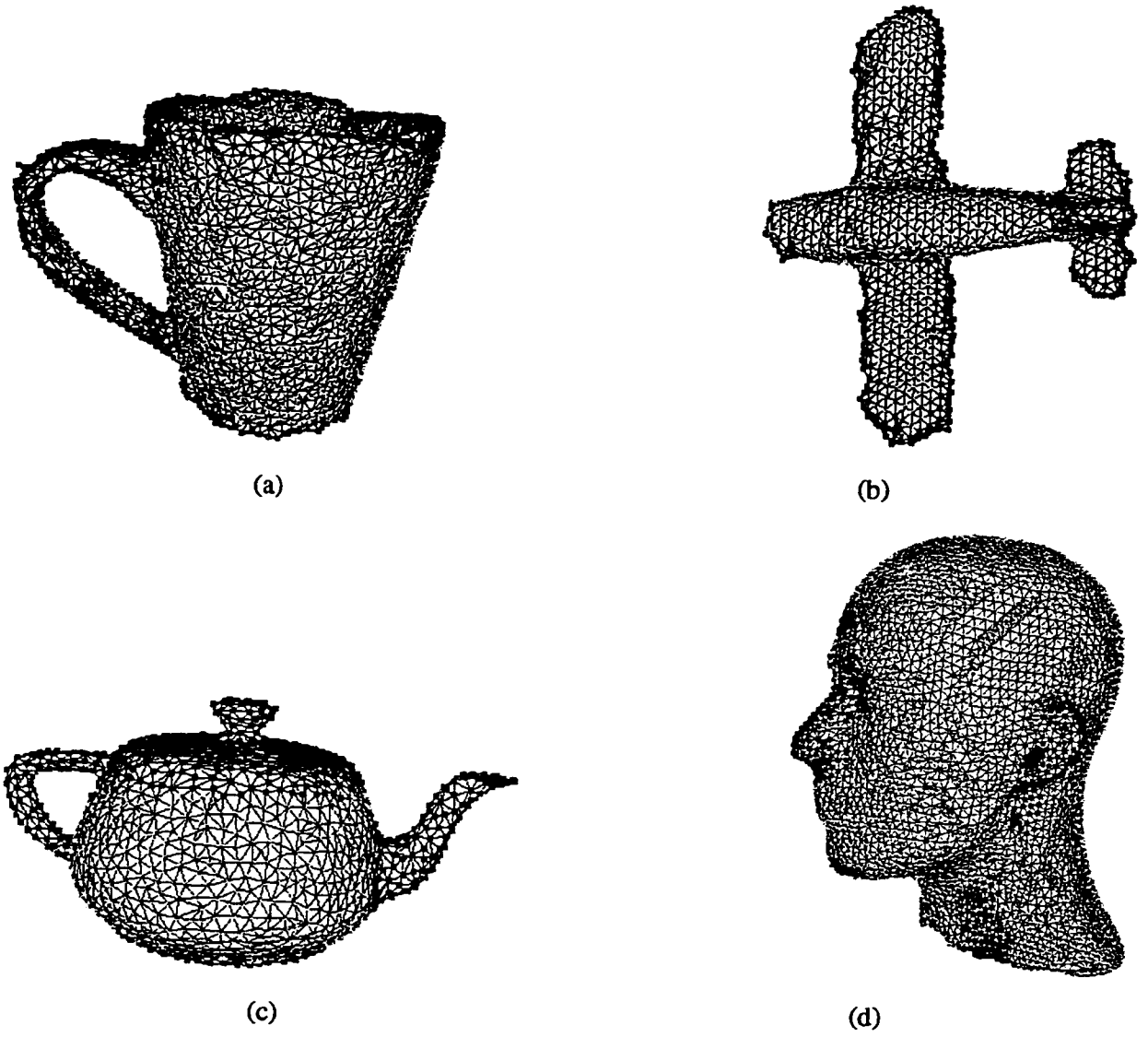
The parameters needed to determine the type of working set to which a vertex belongs are *neighborhood*, *plane_threshold*, and *high_curve_threshold*. A vertex with curvature magnitudes below the plane_threshold will be classified as PLANAR, and one with curvature magnitudes above the high_curve_threshold

will be classified as CONCAVE or CONVEX, depending on the sign of the high curvature. Vertices with curvatures in between the thresholds are classified as OTHER. The neighborhood, N, is dependent upon the mesh resolution; a higher resolution mesh (shorter distance between vertices) will require using neighbors farther away from the vertex. This method of curvature estimation does tend to overestimate curvature, since the Euclidian distance between the vertex and its neighbor is being used, instead of the geodesic distance. We are only interested in relative curvature, though, so this overestimation will not affect the classification of the vertex. The thresholds are also dependent on mesh resolution and N, but we found that once they are set for a particular resolution and N, they did not have to be changed for different objects of the same class.

To obtain the working sets, the normal for each vertex is calculated by fitting a plane to its local neighborhood. Curvature is calculated at each vertex and each is classified according to its type. Regions are grown by grouping together connected vertices of the same type. Small isolated regions that are imbedded in a larger region are then merged into the larger region. The regions are then classified as working sets according to the type of vertices they contain. Figure 4.2 shows several meshes after classification into working sets.

4.4 Shape Types

We developed three allowable shape types that were based on the features present in the object classes we used in our experiments. They are simple shape types, but they are sufficient to illustrate the general nature of our model. They can also be easily adapted to other object classes. The shape types are circle, line segment group, and plane. Powell's method [49] of optimization is used to fit the shape type to the working set by minimizing an error function.



(a) (b)

(c) (d)

Figure 4.2: Working sets for meshes representing (a)Coffee mug, (b) Airplane, (c) Teapot, (d) Mannequin. Red vertices represent CONVEX working sets, blue vertices are CONCAVE working sets, yellow are OTHER working sets and green vertices represent PLANAR working sets

The method of initial estimate generation and the individual error function are shape-dependent. Most initial estimates are based on the *bounding box* of the working set or sets to be fit.

4.4.1 Circle

We used the shape-type *circle* to describe circular separators. Since a circle does not have a unique analytical equation in 3D, but can be described as a circle embedded in a plane, we first fit a plane to the working set using a least squares fit. We then use Powell's method in the four dimensional space of the circle (center coordinates and radius) with the constraint that the circle must stay in the fitted plane, to minimize the distance from the working set points to the circle. The initial estimate for the center of the circle is obtained by projecting the center of the bounding box of the working set to the fitted plane. The initial radius of the circle is half the length of the average of the two largest dimensions of the bounding box. There are eight parameters that completely specify the circle; the planar coefficients A, B, C, D, the circle center x_c, y_c, z_c, and the radius r. Figure 4.3 shows the circle shape-type fit to all CONVEX working sets in the mesh representation of the teapot from Figure 4.2c.

4.4.2 Line Groups

Shape-type *line group* is a general type used to describe features that are piecewise linear. Line groups can consist of any number of line segments with endpoints connected and can be open or closed. Figure 4.4 shows an example of both an open and closed line group with three segments. Figure 4.5 gives an example of a 3-segment fit to the CONVEX working sets in the mesh representation of an airplane from Figure 4.2b. The parameters are the lengths of the line segments formed by each pair of endpoints $\mathrm{ep}_i, \mathrm{ep}_{i+1}$. Powell's method is

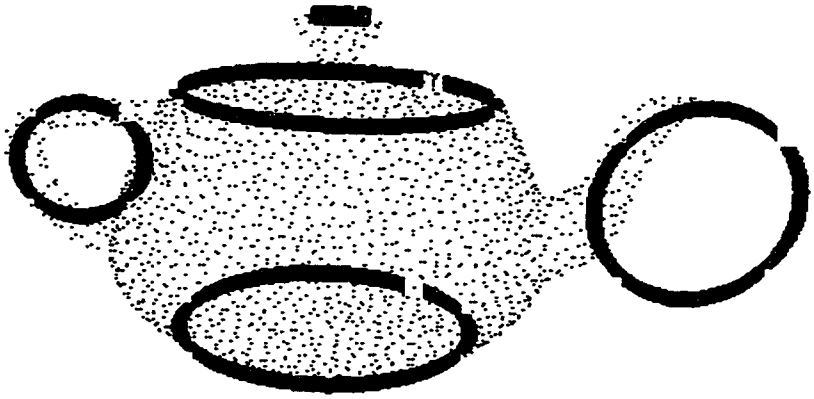


Figure 4.3: Circle shape-type fit to all CONVEX working sets in mesh representation of teapot from Figure 4.2c. Only the mesh vertices are shown here.

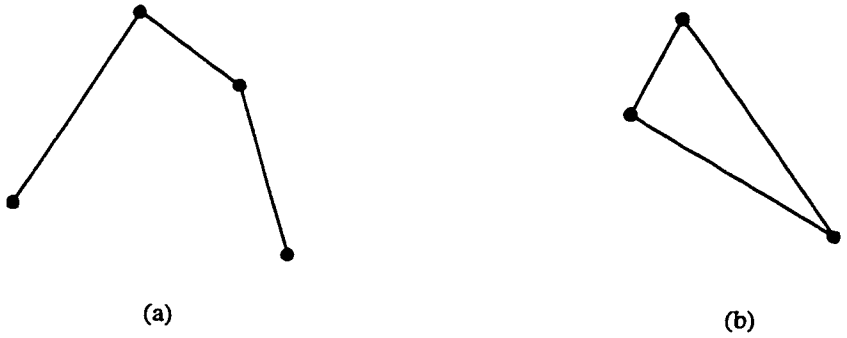


Figure 4.4: Example of an open (a) and a closed (b) line group with 3 segments.

used to adjust the coordinates of the endpoints of the segments so as to minimize the distance from each mesh point p to the line segment nearest p. The initial estimate is determined by the feature parameters and the bounding box with which the feature parameters are associated. The parameters P of a line group LG with N segments are $P(LG) = \{\| \mathbf{ep_0 \cdot ep_1} \|, \ldots, \| \mathbf{ep_{N-1} \cdot ep_N} \|\}$where $\mathbf{ep_i} = (x_i, y_i, z_i)$ for $i = 0, \ldots, N-1$; $\mathbf{ep_i}$ are the endpoints of each line segment. For a closed line group $\mathbf{ep_0} = \mathbf{ep_N}$.

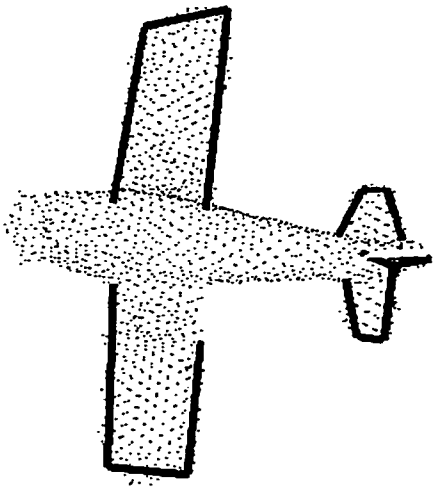


Figure 4.5: Line group shape-type with 3 segments fit to all CONVEX working sets in mesh representation of an airplane from Figure 4.2b. Only the mesh vertices are shown here.

4.4.3 Plane

When we modeled several objects for use in our experiments, we found that there were not enough constraints among the separators to uniquely identify them. For example, on a cup we could identify the top rim and bottom edge, but could not distinguish between the two. We added the surface shape-type PLANE to our allowable shape types to provide sufficient constraints. This illustrates some of the flexibility of our methodology; the working model can model as much or as little of the abstract model as necessary depending on the object and application. The plane fit is done by a simple linear least squares method, so no initial estimate is needed. The parameters of the plane are the coefficients A, B, C and D of the equation for a plane in three dimensions, $Ax + By + Cz + D = 0$. Figure 4.6 shows the the bottom view of the working sets and the plane shape-type fit to the PLANAR working set in the mesh representation of the coffee cup from Figure 4.2a.

Figure 4.6: Bottom view of working sets for coffee cup mesh shown in Figure 4.2a. There is one PLANAR working set. Plane shape-type fit to the PLANAR working set. The plane is square for display purposes only; no assumptions are made regarding its shape.

4.5 Shape Parameters

The shape parameters in the working model are used to both provide initial estimates for the shape fitting to the working set, and to remove those working sets from consideration whose final shape fit does not meet the parameters. The shape parameters had to be general enough to allow variations of features between objects of the same class since our method models an object class, not a particular instance of an object. We found that by using ratios related to the *bounding box* (the smallest box that completely encloses a group of points) of the object, $bb(O)$, or the working set or sets for that constructed feature, $bb(w)$, we could specify parameter ranges that were scale invariant. Each parameter has a minimum and maximum value. These values are automatically calculated at run-time for the individual instance of the class.

4.6 *Relationships*

The relationships we instantiated for our experiments are *parallel*, *perpendicular*, *reflective symmetry*, *centroid_distance*, and *concentric*. The relationships are defined for the allowed shape-types as follows:

- Parallel– Feature f_1 is *parallel to* feature f_2 if, for best-fit plane $A_1x + B_1y + C_1z + D_1 = 0$ to f_1, and $A_2x + B_2y + C_2z + D_2 = 0$ to f_2, and normals $\mathbf{F_{1N}} = (A_1, B_1, C_1)$, $\mathbf{F_{2N}} = (A_2, B_2, C_2)$, $|\mathbf{F_{1N}} \cdot \mathbf{F_{2N}}| \approx 0$. See Figure 4.7.

- Perpendicular–Feature f_1 is *perpendicular to* feature f_2 if, for best-fit plane $A_1x + B_1y + C_1z + D_1 = 0$ to f_1, and $A_2x + B_2y + C_2z + D_2 = 0$ to f_2, and normals $\mathbf{F_{1N}} = (A_1, B_1, C_1)$, $\mathbf{F_{2N}} = (A_2, B_2, C_2)$, $|\mathbf{F_{1N}} \cdot \mathbf{F_{2N}}| \approx 1$. Figure 4.8 illustrates the perpendicular relationship.

- Reflective Symmetry– Features f_1 and f_2 have *reflective symmetry* if there exists a plane RP that for every point p in f_1 there is a corresponding point q in f_2 such that the midpoint, $\frac{(p+q)}{2}$, lies in RP. The line segments p_iq_i must be perpendicular to RP. See Figure 4.9 for an example of reflective symmetry.

- Centroid_Distance(min,max)– This relationship is the distance between the centroids of two features, and has parameters *min* and *max* associated with it. The centroid is defined for each allowable shape type. The constraint relationship is met if the distance d between the centroids of the fitted functions falls within the range (min, max). This range is based on the bounding box of the object and calculated at run time. See Figure 4.10.

- Concentric–Only for circular features, planes must meet the *parallel* constraint and the line segment formed by the centers of the circles must

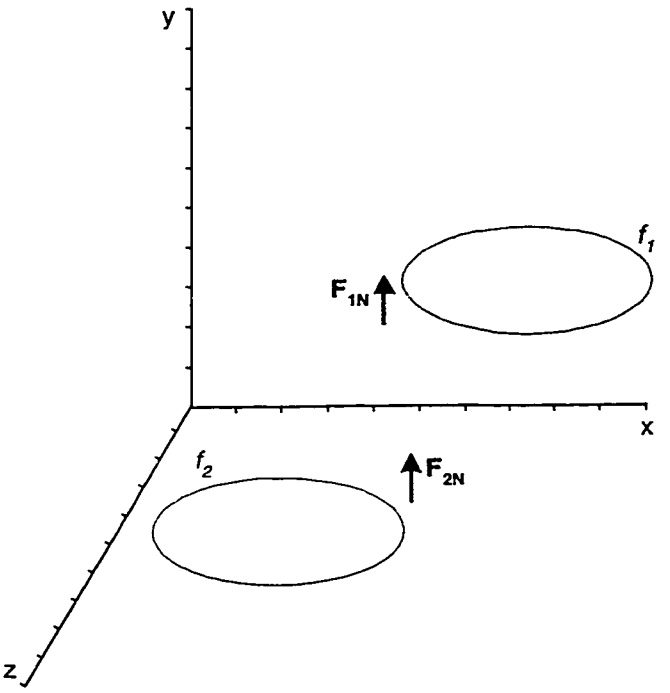


Figure 4.7: Example of the *parallel* relationship. Features f_1 and f_2 are circles embedded in planes with normals F_{1N} and F_{2N} respectively. The two circles are parallel because the dot product of F_{1N} and F_{2N} is 0.

be perpendicular to the planes of the circle. Figure 4.11 shows a pair of concentric circles.

4.7 Summary and Contributions

In this chapter we have discussed surface curvature characteristics and developed four working set types based on local surface curvature. We defined three geometric shape types and illustrated shape fit to the working sets. Finally, we formulated five constraint relationships between the features for use in our prototype system. The major contribution of this chapter is the implementation of the working model described in the previous chapter.

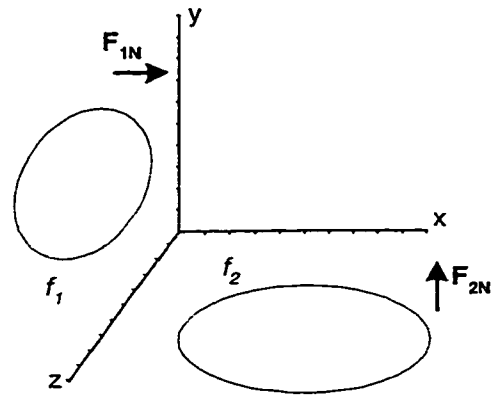


Figure 4.8: Example of the *perpendicular* relationship. Features f_1 and f_2 are circles embedded in planes with normals $\mathbf{F_{1N}}$ and $\mathbf{F_{2N}}$ respectively. The two circles are perpendicular because the magnitude of the dot product of $\mathbf{F_{1N}}$ and $\mathbf{F_{2N}}$ is 1.

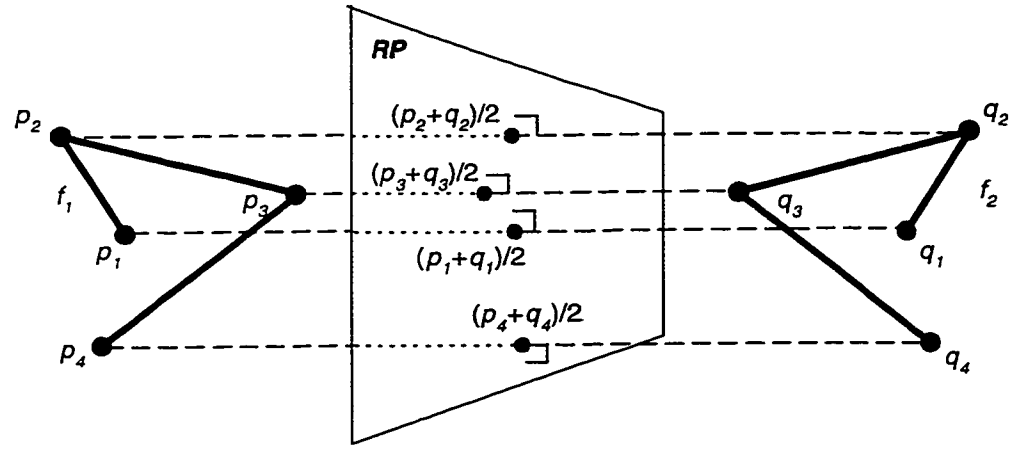


Figure 4.9: Example of *reflective symmetry*. Line group features f_1 and f_2 have reflective symmetry with respect to plane RP. The indicated line segments are bisected by RP and are also perpendicular to RP

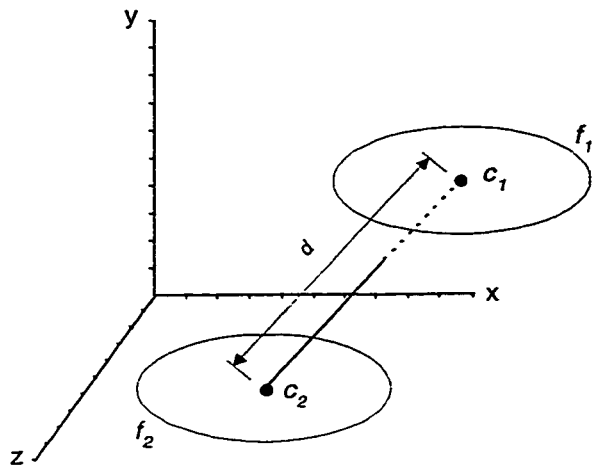


Figure 4.10: Example of *centroid distance*. The points c_1 and c_2 are the centroids of features f_1 and f_2. The centroid distance d is the Euclidian distance between the two points.

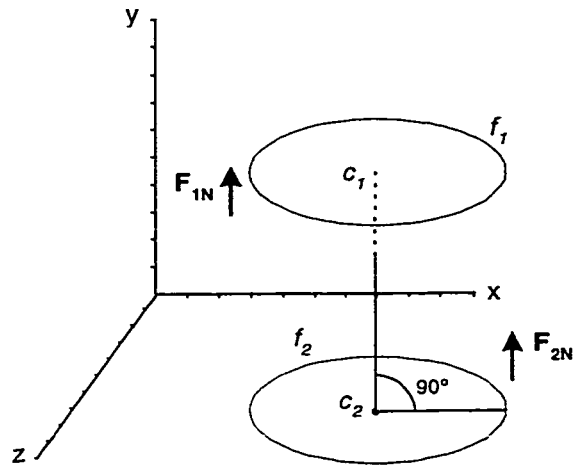


Figure 4.11: Example of *concentric* relationship. Features f_1 and f_2 are parallel as in Figure 4.7, but in addition the line segment c_1c_2 is perpendicular to both features.

Chapter 5

EXPERIMENTAL DESIGN

5.1 Introduction

To test our prototype system, we implemented three working models. This chapter is a discussion of those models and the experiments we performed. The object classes of the models are pipe fitting, cup and airplane. We experimented with five instances of pipe fitting, five instances of cup, and three instances of airplane. For each instance of cup and pipe we used three different mesh resolutions. We performed both qualitative and quantitative analyses on the meshes.

5.2 Working Models

In our working models for these objects, we used three of the four working set types: convex, concave, and planar. We used all three shape types, and will use the following conventions when describing each model. Shape-type *line group* is represented by $lgrpx$, where x is an integer representing the number of segments. The relationship *centroid_distance* is abbreviated $cdist(min, max)$. Recall that the *shape parameters* allow us to specify ranges for the fitted shapes based on the bounding box of either the object or the working set(s). The bounding box conventions we used in implementing our working model are as follows: $bb(O)$ means the bounding box of the entire object, $bb(w)$ means the bounding box of the working set or sets. On the bounding boxes themselves, a is the longest dimension, b is the next longest dimension and c is the shortest dimen-

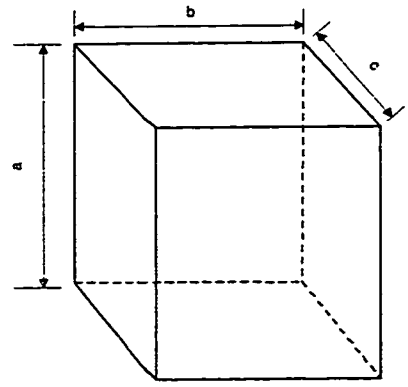


Figure 5.1: Convention for measurement of bounding box. "a" is the largest dimension, "b" is second largest, and "c" is the smallest dimension of a bounding box.

sion. A parameter such as $\frac{bb(0):a}{2}$ means "half the length of the longest side of the bounding box of the object." See Figure 5.1.

5.2.1 Coffee Cup Working Model

The coffee cup has four features, which are rim, bottomEdge, bottom, and handleCon. See Figure 5.2 for a sketch of the features. Table 5.1 shows each feature with its associated properties and the relationships between the features. The detectabilities were obtained experimentally, and the shape parameters were obtained by observation of many instances of the class. The "rim" feature is the top rim of the cup. It has a convex working set and is circular in shape. The shape parameters state that the radius of the fitted circle must be no smaller than half the size of the smallest dimension of the working set bounding box, and no larger than half the size of the largest dimension of the object bounding box. If a working set happens to be fairly linear in shape and is fit with a circle, the radius becomes huge. This somewhat broad range of shape parameters bounds the circle to the size of the bounding box. The "bottomEdge" feature is where the bottom surface of the cup meets the side of the cup. It is also circular

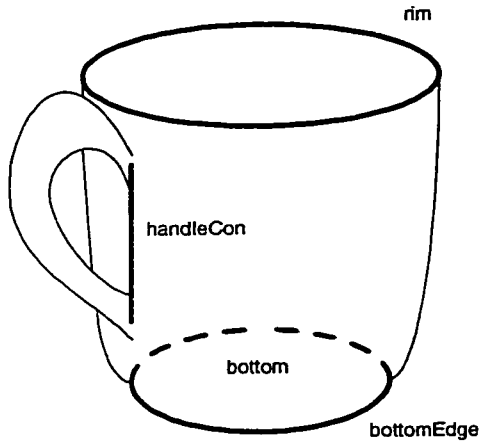


Figure 5.2: Features of a coffee cup.

in shape, and the shape parameters are the same as those for the rim. The "bottom" is the bottom surface of the cup, which is planar and has no restrictions on the shape parameters. The last feature, "handleCon," is a line segment between the top and bottom connections of the handle to the body of the cup. This feature is formed from two concave working sets. The shape parameters state that the distance between the endpoints of the line segment must be no larger than the largest dimension of the object bounding box, and no smaller than one-fourth the largest dimension of the object bounding box.

There are four relationships defined between the features of the cup. They are are follows:

1. The rim and edge must be concentric.

2. The distance between the bottomEdge and the bottom must be less than one-fourth the smallest dimension of the bounding box of the object.

3. The handleCon must be perpendicular to the rim.

4. The handleCon must be perpendicular to the bottomEdge.

Relationships 3 and 4 are redundant to ensure that if either the rim or bottomEdge was not found there would still be a constrained relationship between handleCon and the other feature.

5.2.2 Airplane Working Model

We developed a partial working model of the airplane to illustrate different shape types and relationships than those shown in the coffee cup. The airplane is a much more complex object than the coffee cup and full development of the working model is beyond the scope of our prototype system.

The working model of the airplane has five features. The port wing edge *(pWingE)* and starboard wing edge *(sWingE)* are the edges where the top and bottom surfaces intersect on the respective left and right wings. The port horizontal stabilizer edge *(pHzStabE)* and starboard horizontal stabilizer edge *(sHzStabE)* are the edges where the top and bottom surfaces intersect on the left and right horizontal stabilizers, respectively. The vertical stabilizer edge *(vStabE)* is where the port and starboard surfaces intersect on the vertical stabilizer. See Figure 5.3 for a sketch of the features. All constructed features in the working model are formed from convex working sets and all are fit with the shape type *line group* using 3 segments, represented by *lgrp3*.

Table 5.2 shows each feature with its associated parameters and Table 5.3 shows the relationships between the features. Since all features are the same general shape, the shape parameters are the same with respect to the bounding box of the working set. As with the coffee cup, detectabilities were obtained experimentally and shape parameters were obtained by observing the characteristics of the lift surfaces of various aircraft. The shape parameters state that the first and third line segments should be no longer than the largest di-

Table 5.1: Working model of the coffee cup. The parameter $bb(O) : a$ means "the largest dimension of the bounding box of the object, while $\frac{bb(w):c}{2}$ means "half the length of the shortest dimension of the bounding box enclosing the working sets." The parameters $\mathbf{ep_0}, \mathbf{ep_1}$ are the endpoints of the line segment forming the shape-type *lgrp1*.

WORKING MODEL OF CUP

Features				
feature label	*working_set type*	*shape type*	*shape parameters*	*detectability*
rim	convex	circle	rim_parms	1.0
bottomEdge	convex	circle	botm_parms	0.9
bottom	planar	plane	none	1.0
handleCon	concave,concave	lgrp1	hndl_parms	0.5

rim_parms= $\frac{bb(w):c}{2} < radius < bb(O) : a$

botm_parms= $\frac{bb(w):c}{2} < radius < bb(O) : a$

hndl_parms= $\frac{bb(O):a}{4} <\| \mathbf{ep_0} \cdot \mathbf{ep_1} \| < bb(O) : a$

Relationships $R = \{r_1, r_2, r_3, r_4\}$

$r_1 = (rim, botmEdge, concentric)$

$r_2 = (botmEdge, bottom, cdist(0, \frac{bb(O):c}{4}))$

$r_3 = (handleCon, rim, perpendicular)$

$r_4 = (handleCon, bottomEdge, perpendicular)$

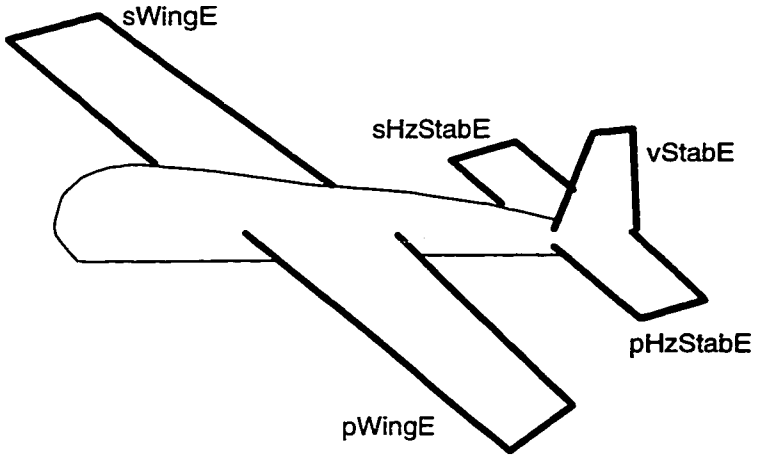


Figure 5.3: Features of the airplane working model. The port and starboard wing edges are *pWingE* and *sWingE*, respectively. The vertical stabilizer edge is *vStab*, and the port and starboard horizontal stabilizers are *pHzStab* and *sHzStab*.

mension of the working set bounding box, and the center segment should be no longer than the middle dimension of the working set bounding box. There are eight relationships between the features of the airplane. They are:

1. The port wing edge has reflective symmetry with the starboard wing edge.

2. The port wing edge is parallel to the starboard wing edge.

3. The distance between the centroids of the port and starboard wing edges is no smaller than one-fourth the middle dimension of the object bounding box and no larger than the middle dimension of the object bounding box.

4. The port horizontal stabilizer has reflective symmetry with the starboard horizontal stabilizer.

5. The vertical stabilizer is perpendicular to the port horizontal stabilizer.

6. The vertical stabilizer is perpendicular to the starboard horizontal stabilizer.

7. The centroid distance between the vertical stabilizer and the port horizontal stabilizer is less than the smallest dimension of the object bonding box.

8. The centroid distance between the vertical stabilizer and the starboard horizontal stabilizer is less than the smallest dimension of the object bonding box.

Relationships 5 and 6 are redundant, as are relationships 7 and 8. The centroid of a line group with N segments for $N \geq 2$ is defined as the center of the bounding box enclosing the endpoints of all segments.

This partial model does not differentiate between the port and starboard wings or the port and starboard horizontal stabilizers. The spatial associations network of the abstract model would perform this task, as it is the location relative to the nose and tail that distinguish the port and starboard side of an aircraft.

5.2.3 Pipe Fittings Working Model

The purpose of the pipe fittings is to explore a rudimentary method of object recognition using our system. Rather than having a separate working model for each type of pipe fitting, we use a single working model with all possible relationships and features. The features and relationships found will determine the type of pipe fitting the mesh represents. We used five different pipe fittings as shown in Figure 5.4. The main feature of all the pipe fittings is the circular openings, or rims. It is not possible to tell the difference between one rim and

Table 5.2: Features and parameters for the working model of the airplane. The ep_i are the endpoints of the line segments forming the shape-type *lgrp3*.

WORKING MODEL OF AIRPLANE

Features				
feature label	*working_set type*	*shape type*	*shape parameters*	*detectability*
pWingE	convex	lgrp3	pWingE_parms	1.0
sWingE	convex	lgrp3	sWingE_parms	1.0
pHzStabE	convex	lgrp3	pHzStabE_parms	0.8
sHzStabE	convex	lgrp3	sHzStabE_parms	0.8
vStab	convex	lgrp3	vStab_parms	0.5

$$\text{pWingE\_parms} = (0 <\| \mathbf{ep_0} \cdot \mathbf{ep_1} \|< bb(w) : a,\ 0 <\| \mathbf{ep_1} \cdot \mathbf{ep_2} \|< bb(w) : b,\ 0 <\| \mathbf{ep_2} \cdot \mathbf{ep_3} \|< bb(w) : a)$$

$$\text{sWingE\_parms} = (0 <\| \mathbf{ep_0} \cdot \mathbf{ep_1} \|< bb(w) : a,\ 0 <\| \mathbf{ep_1} \cdot \mathbf{ep_2} \|< bb(w) : b,\ 0 <\| \mathbf{ep_2} \cdot \mathbf{ep_3} \|< bb(w) : a)$$

$$\text{pHzStabE\_parms} = (0 <\| \mathbf{ep_0} \cdot \mathbf{ep_1} \|< bb(w) : a,\ 0 <\| \mathbf{ep_1} \cdot \mathbf{ep_2} \|< bb(w) : b,\ 0 <\| \mathbf{ep_2} \cdot \mathbf{ep_3} \|< bb(w) : a)$$

$$\text{sHzStabE\_parms} = (0 <\| \mathbf{ep_0} \cdot \mathbf{ep_1} \|< bb(w) : a,\ 0 <\| \mathbf{ep_1} \cdot \mathbf{ep_2} \|< bb(w) : b,\ 0 <\| \mathbf{ep_2} \cdot \mathbf{ep_3} \|< bb(w) : a)$$

$$\text{vStabE\_parms} = (0 <\| \mathbf{ep_0} \cdot \mathbf{ep_1} \|< bb(w) : a,\ 0 <\| \mathbf{ep_1} \cdot \mathbf{ep_2} \|< bb(w) : b,\ 0 <\| \mathbf{ep_2} \cdot \mathbf{ep_3} \|< bb(w) : a)$$

Table 5.3: Relationships for the working model of the airplane.

WORKING MODEL OF AIRPLANE *(cont.)*

Relationships $R = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7\}$

$r_1 = (pWingE, sWingE, refl\_sym)$

$r_2 = (pWingE, sWingE, parallel)$

$r_3 = (pWingE, sWingE, cdist(\frac{bb(O):b}{4}, bb(O) : b))$

$r_4 = (pHzStabE, sHzStabE, refl\_sym)$

$r_5 = (pHzStabE, vStabE, perpendicular)$

$r_6 = (sHzStabE, vStabE, perpendicular)$

$r_7 = (pHzStabE, vStabE, cdist(0, bb(O) : c))$

$r_8 = (sHzStabE, vStabE, cdist(0, bb(O) : c))$

another except by the relationship between them. We fit the shape type *circle* to convex working groups for all types of pipe fittings. The *wye* fitting has two parallel circles and a third circle whose center is near one of the parallel circles. The *tee* fitting has two parallel circles and a third that is perpendicular to the other two. The *cross* fitting has four circles; two parallel pairs that are perpendicular to each other. The *elbow* fitting has two perpendicular circles. The *tricorner* fitting has three circles, all of which are perpendicular to one another. Table 5.4 shows the working model for the pipe fittings. Detectabilities were determined by the likelihood that a given feature will be present. Rim1 and rim2 are present in all fittings, so detectability is 1.0. Rim3 is present in four of the five fittings so its detectability is 0.8. Since rim4 occurs in only one out of five fittings, the detectability is 0.2. The features are all the same; it is the relationships that distinguish one type of fitting from another. Not all relationships will be satisfied; some, such as r_1 and r_2, are even mutually exclusive.

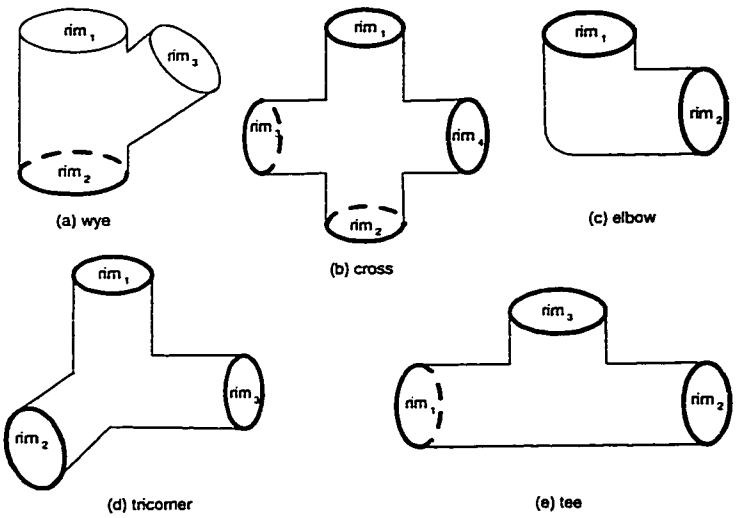


Figure 5.4: Five different pipe fittings and their features.

The relationships are tested and the combination of satisfied relationships is used to determine the type of fitting represented by the mesh. Table 5.5 shows the combinations of relationships that must be satisfied for each type of pipe fitting.

5.3 Experiments

The experiments consist of two components: a qualitative part and a quantitative part. Three separate classes of objects were used, with a total of thirteen different object instances. Mesh resolutions of twenty thousand, ten thousand, and five thousand faces for each object of class "cup" and class "pipe fitting" were used. Details of the experiments and the results are discussed in the next chapter.

Table 5.4: Working model of the pipe fittings. Not all relationships can be satisfied. Type of fitting is determined by which relationships are satisfied.

WORKING MODEL OF PIPE FITTINGS

Features				
feature label	*working_set type*	*shape type*	*shape parameters*	*detectability*
rim1	convex	circle	rim_parms	1.0
rim2	convex	circle	rim_parms	1.0
rim3	convex	circle	rim_parms	0.8
rim4	convex	circle	rim_parms	0.2

rim_parms= $\frac{bb(w):c}{2} < radius < bb(O) : a$

Relationships $R = \{r_1, r_2, r_3, r_4, r_5\}$

$r_1 = (rim1, rim2, parallel)$

$r_2 = (rim1, rim2, perpendicular)$

$r_3 = (rim1, rim3, cdist(0, \frac{bb(O):a}{2}))$

$r_4 = (rim1, rim3, perpendicular)$

$r_5 = (rim3, rim4, parallel)$

Table 5.5: Relationships satisfied by pipe fitting type. Bullet in column indicates relationship must be satisfied.

Relationships Satisfied by Pipe Fitting Type						
	Relationship					
		r_1	r_2	r_3	r_4	r_5
Type of Pipe Fitting	wye	•		•		
	cross	•			•	•
	elbow		•			
	tricorner		•		•	
	tee	•			•	

5.3.1 Qualitative Testing

The qualitative part of our testing compared features present, features labeled correctly, features present but not labeled, and features labeled incorrectly. The mesh was displayed using 3-D graphics so it could be rotated and scaled interactively. A human operator observed the mesh and determined which features were present. Our system was then used to label and display the features, and the operator determined whether or not the features were labeled correctly.

5.3.2 Quantitative Testing

The quantitative part of testing took a human operator's input as "ground truth" and calculated average distance from the constructed feature to the actual feature. The mesh was displayed and a human operator marked several vertices as belonging to a particular feature. Our system was then used to label the features, and the average distance from the constructed feature to the

operator-labeled vertices was calculated.

5.4 Summary and Contributions

In this chapter, we developed detailed working models for the objects used in our experiments. We described the features of interest in each model, and discussed the relationships between the features. For the pipe fittings, the relationships were examined to determine classification of the object. The contribution of this chapter is the development and detailed description of working models for three different classes of objects.

Chapter 6

EXPERIMENTS AND RESULTS

6.1 Introduction

This chapter describes our experiments in detail and discusses the results of the testing, including both qualitative and quantitative analyses.

6.2 Experiments and Results

6.2.1 Coffee Cup

We generated five instances of the coffee cup class. Each cup has different shape variations which help test our theory of a generic coffee cup feature finder. Figure 6.1 shows an intensity image of each coffee cup along with the dense mesh ($\sim$ 20K faces) generated by our system. The colored patterns on the objects are used to satisfy the requirements of Pulli's registration algorithm, which is used in mesh construction. Our work does not use any information from intensity images. Figures 6.2 through 6.16 show the processing stages for all of the coffee cups. Part (a) is the normalized mesh. Part (b) shows the working sets obtained from the bottom up stage of processing. Constructed features are shown in part (c), with constructed features for the rim and bottom edge shown in red, bottom surface constructed features shown in blue, and handle constructed features shown in orange. Part (d) shows the final feature labeling. The rim feature is indicated in green, the bottom edge in magenta, the handle in orange, and the bottom surface in cyan. The shape of the bottom surface

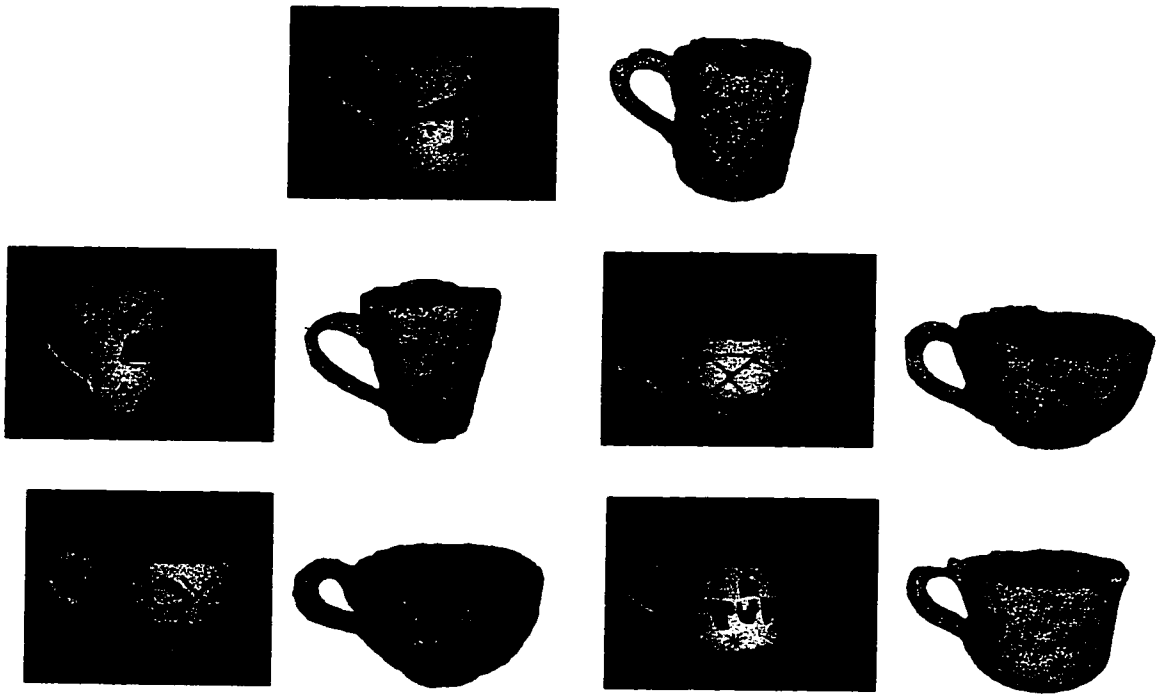


Figure 6.1: The five instances of "coffee cup" used in experiments. On the left is the intensity image of one view of each cup, while on the right is a mesh constructed by our system after normalization and reduction to approximately 20K triangles .

is artificial. To display the infinite plane fit to the surface, we made artificial boundaries using the bounding box of the fitted region. Part (e) shows points manually marked on the mesh by a human operator before processing, which will be used in the quantitative results section.

6.2.2 Qualitative Results for Coffee Cup

Our system reliably labeled the correct features in most instances of the coffee cup. For cup 1 and cup 2 (Figures 6.2 - 6.7) mesh resolution did not seem to affect the results. Different resolutions caused different constructed features

to be generated, but the correct constructed feature was always generated and properly labeled.

For cup 3, mesh resolution made a great deal of difference. In the 5K instance shown in Figure 6.8, the labeled features are in the correct relationships to each other and in the correct general area of the coffee cup, but the constructed features associated with those labels do not fit the manually marked mesh points very well. This is caused by the smoothing of the area where the top of the handle meets the body of the cup. The area is not sufficiently concave to be detected during the region-growing phase so there is no concave working set there. This causes the convex part of the handle to be grown into the convex part of the rim, forming one convex working set, where there should be at least two. The circle fit to this working set during constructed feature generation resembles the rim, but is skewed toward the handle. Part of the inside of the handle is sufficiently concave to form a working set, thus causing a constructed feature to be formed between that area and the bottom of the handle. The bottom edge of cup 3 is also affected by the mesh smoothing. The actual bottom of this particular cup has a thick edge that is not as well defined as cups 1 and 2. Only two small regions at the bottom are convex, as shown by the two constructed features on the bottom of the cup in Figure 6.8c. The larger circle does meet the constraints, but does not fit the manually marked points very well.

The 10K instance of cup 3, shown in Figure 6.9, indicates a marked improvement in the constructed features. The concave region at the top of the handle is now present, so the convex rim and handle regions are two different working sets. More convex points are present at the bottom edge causing a better circle fit to that region. The concave region at the top of the handle is grown into the concave region on the inside of the handle, so the handle feature is not as close to the marked feature points as we would like. In Figure 6.10, the 20K instance of cup3, the concave regions in the handle have been separated, as indicated by

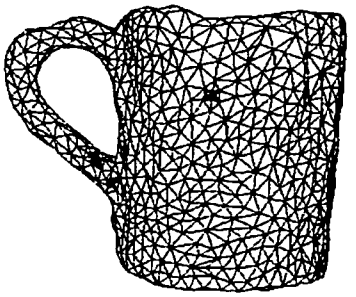
the presence of three constructed features for the handle feature. The correct constructed feature is labeled as the handle feature.

In Figures 6.11 - 6.13, the results for cup 4, we can observe the same problem with the handle feature as we saw in cup 3. The 5K mesh is not a sufficient resolution to detect concave points at the top of the handle. In this case, however, this problem causes the handle feature not to be found. The 10K and 20K meshes do not exhibit this problem, but in the 10K mesh the convex regions of the handle and rim are grown together causing the constructed feature at the rim to be skewed. The 20K mesh finds all features and the constructed features are very close to the marked points.

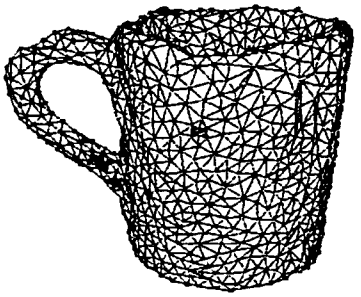
Cup 5, shown in Figures 6.14 - 6.16, presents an interesting problem due to the unusual shape of the handle. It was somewhat difficult even for a human operator to decide exactly where the handle met the body of the cup at the top. This cup also had an unusual bottom edge, and at the 5K resolution, the human operator had problems determining exactly where the bottom edge was. The constructed features in Figure 6.14c indicate some of the difficulties, especially with the bottom edge. Some improvement is seen in most constructed features in the 10K mesh (Figure 6.15), though the rim feature is skewed toward the handle. The 20K mesh shown in Figure 6.16 has correct constructed features that are correctly labeled.

6.2.3 Quantitative Results for Coffee Cup

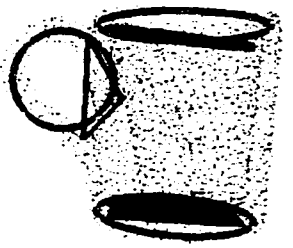
Quantitative analyses of the results of our system was a challenge, because we have no measure of "ground truth." While we would like to compare our fitted features with actual points on the object, we have no perfect representation of the object. We have instead an imperfect reconstruction of an object in the form of a triangular mesh. In the absence of ground truth, we compared our constructed features to the points a human would perceive as the actual



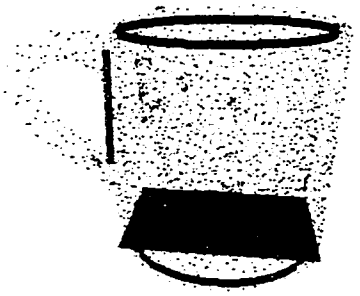
(a) Normalized mesh



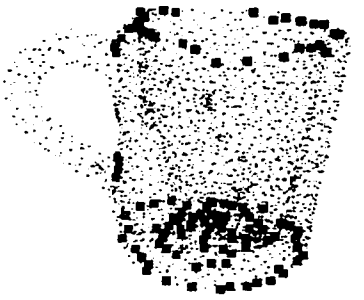
(b) Working sets



(c) Constructed features



(d) Found features



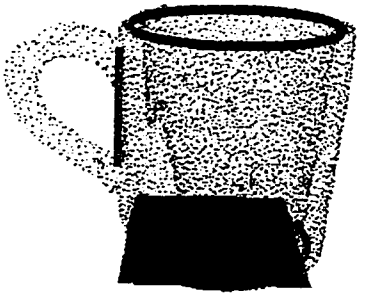
(e) Manually marked points

Figure 6.2: Processing stages for cup 1 with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

(a) Normalized mesh

(b) Working sets

(c) Constructed features



(d) Found features

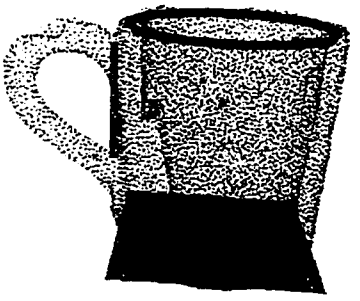
(e) Manually marked points

Figure 6.3: Processing stages for cup 1 with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

(a) Normalized mesh

(b) Working sets

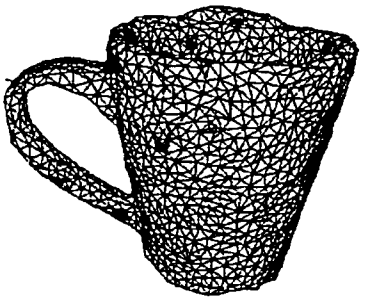
(c) Constructed features



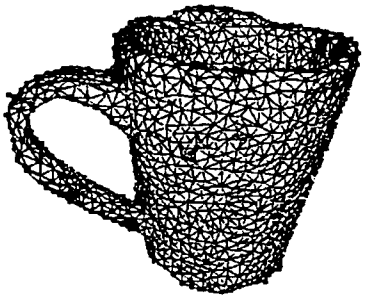
(d) Found features

(e) Manually marked points

Figure 6.4: Processing stages for cup 1 with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

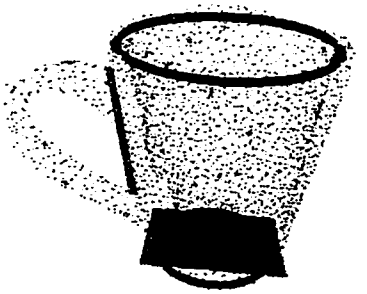


(a) Normalized mesh

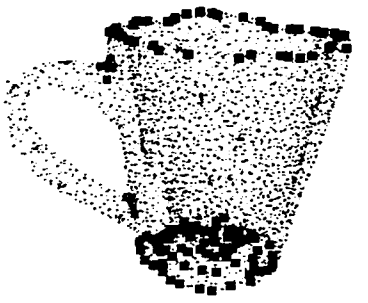


(b) Working sets

(c) Constructed features

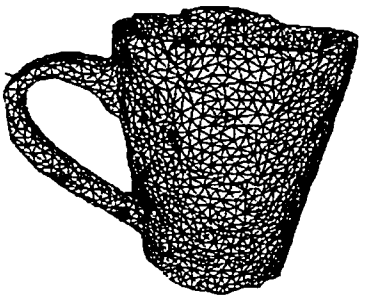


(d) Found features

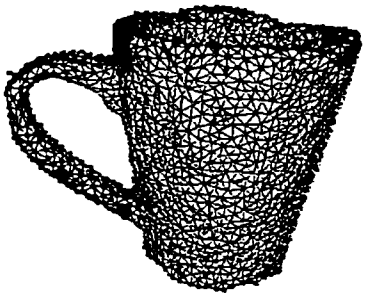


(e) Manually marked points

Figure 6.5: Processing stages for cup 2 with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

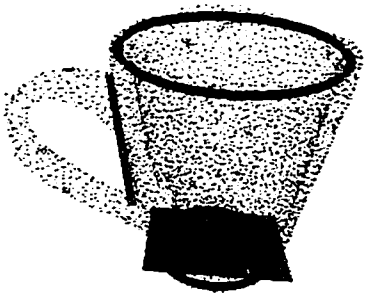


(a) Normalized mesh



(b) Working sets

(c) Constructed features



(d) Found features

(e) Manually marked points

Figure 6.6: Processing stages for cup 2 with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

(a) Normalized mesh

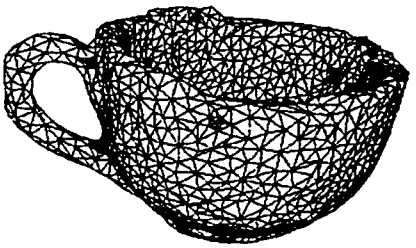
(b) Working sets

(c) Constructed features

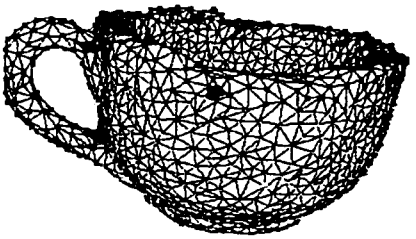
(d) Found features

(e) Manually marked points

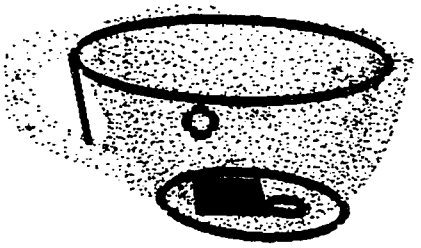
Figure 6.7: Processing stages for cup 2 with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.



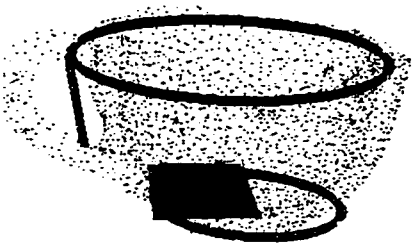
(a) Normalized mesh



(b) Working sets



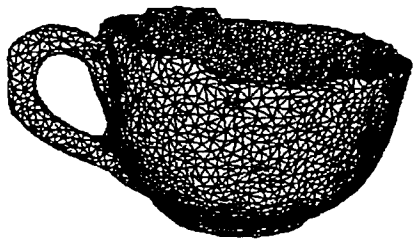
(c) Constructed features



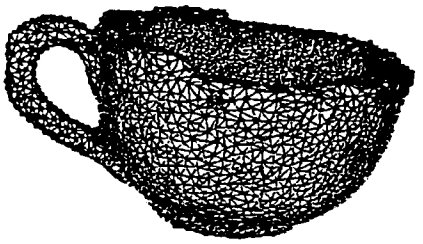
(d) Found features

(e) Manually marked points

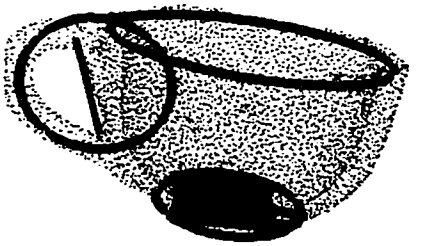
Figure 6.8: Processing stages for cup 3 with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.



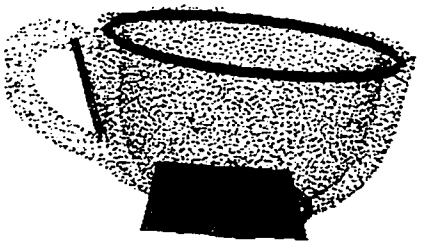
(a) Normalized mesh



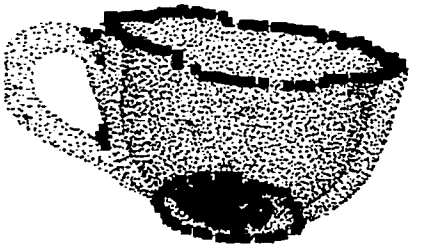
(b) Working sets



(c) Constructed features



(d) Found features

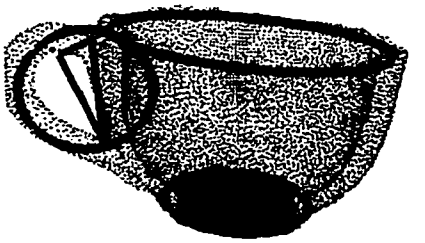


(e) Manually marked points

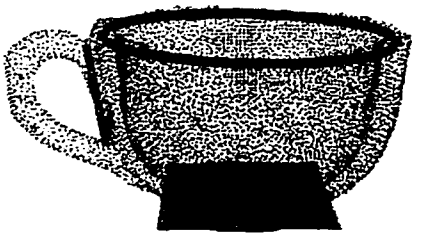
Figure 6.9: Processing stages for cup 3 with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

(a) Normalized mesh

(b) Working sets



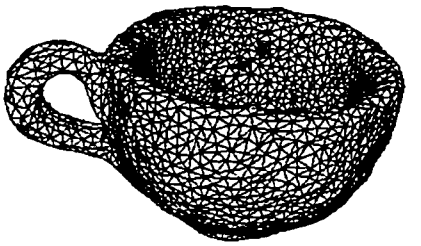
(c) Constructed features



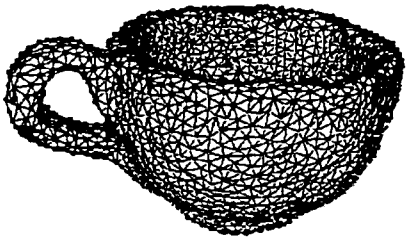
(d) Found features

(e) Manually marked points

Figure 6.10: Processing stages for cup 3 with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.



(a) Normalized mesh



(b) Working sets

(c) Constructed features

(d) Found features

(e) Manually marked points

Figure 6.11: Processing stages for cup 4 with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

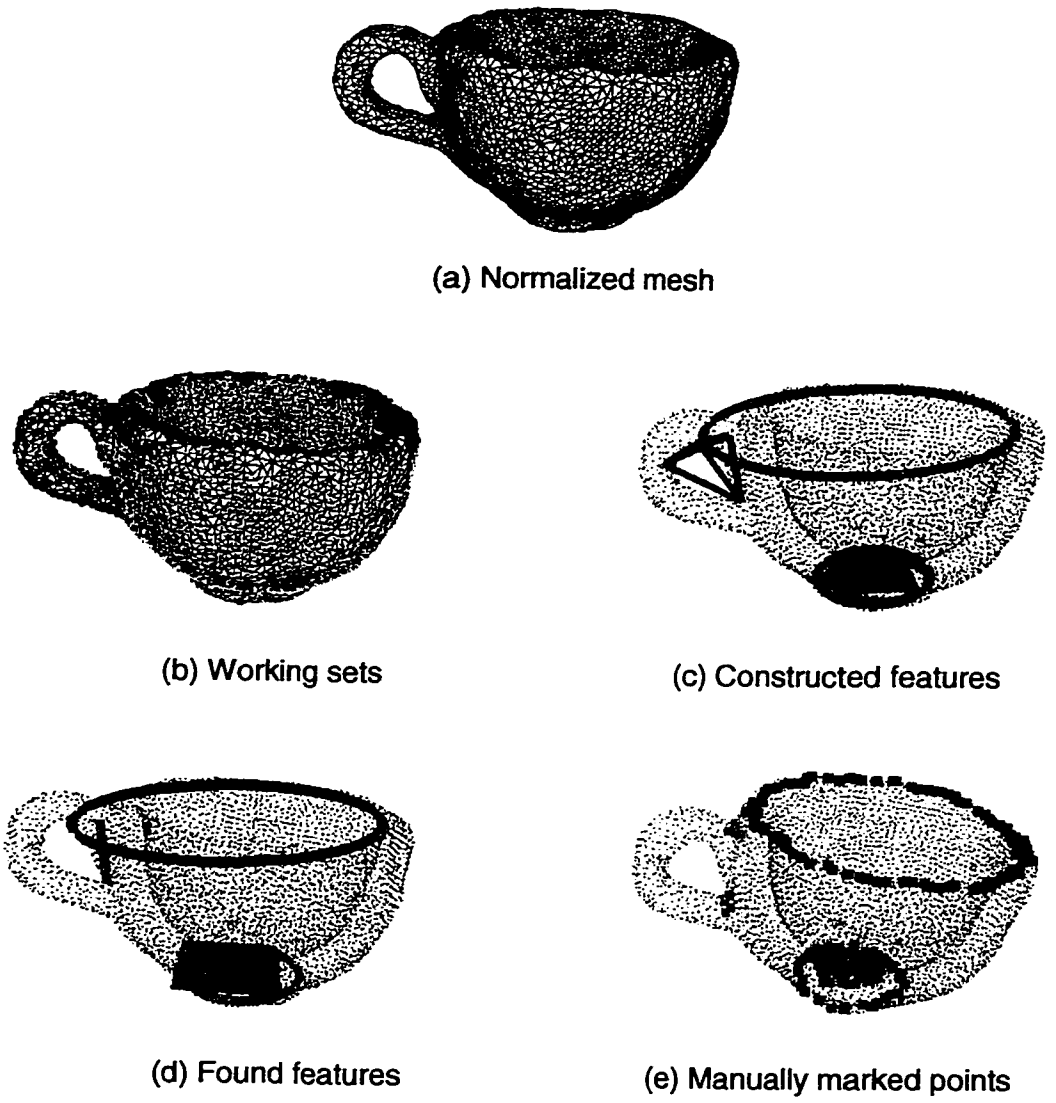
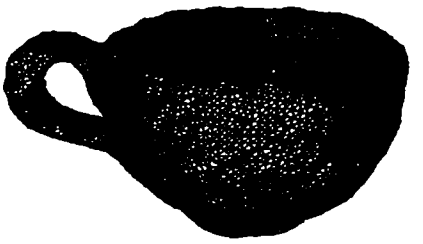


Figure 6.12: Processing stages for cup 4 with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.



(a) Normalized mesh

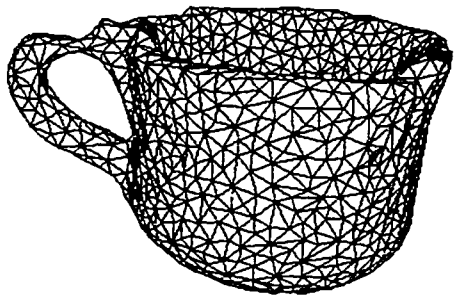
(b) Working sets

(c) Constructed features

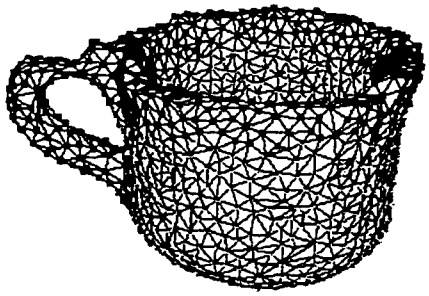
(d) Found features

(e) Manually marked points

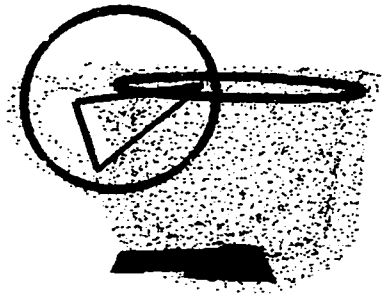
Figure 6.13: Processing stages for cup 4 with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.



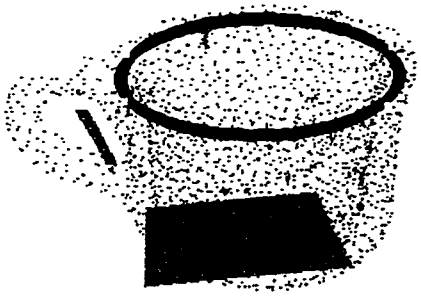
(a) Normalized mesh



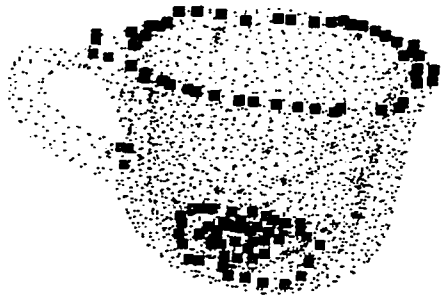
(b) Working sets



(c) Constructed features

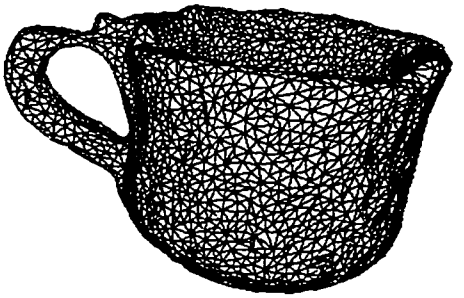


(d) Found features

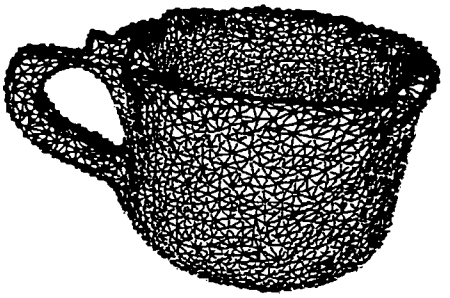


(e) Manually marked points

Figure 6.14: Processing stages for cup 5 with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

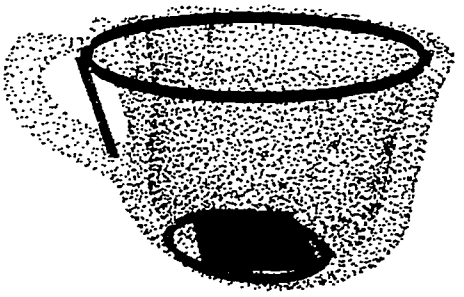


(a) Normalized mesh

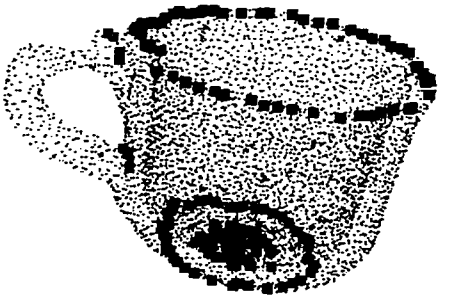


(b) Working sets

(c) Constructed features



(d) Found features

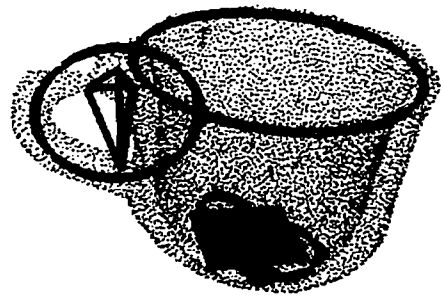


(e) Manually marked points

Figure 6.15: Processing stages for cup 5 with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

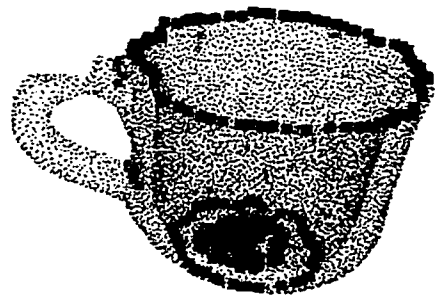
(a) Normalized mesh

(b) Working sets



(c) Constructed features

(d) Found features



(e) Manually marked points

Figure 6.16: Processing stages for cup 5 with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are shown in red, planar constructed features are shown in blue, and line group constructed features in orange. Found features are shown in (d). The rim is green, bottom edge is magenta, bottom surface plane is cyan, and the handle feature is orange. Part (e) shows the manually marked points used in quantitative testing.

features, given the imperfect mesh.

The quantitative results are found in Table 6.1. These results were obtained by calculating the closest distance from each marked point in part (e) of Figures 6.2 through 6.16 to its corresponding labeled constructed feature shown in part (d) of the same figures. Table 6.1 shows the average distance in millimeters. These distances do not indicate the amount of error from actual locations on the coffee cup; they could be caused by errors in the mesh as much as errors in our feature fit. We can, however, use them to indicate how well our function fits the region in the mesh that a human would perceive as that feature. For example, it is visually obvious that the rim on cup 4 at 10K resolution is skewed from the actual rim (Figure 6.12d), and at 20K resolution has a much better fit (Figure 6.13d). This is shown in the quantitative results in Table 6.1 as a larger distance from manually marked features in the 10k case compared to the distance in the 20K case.

6.2.4 Airplane

We used one synthetic and two real instances of the airplane. The input meshes were not good enough to test our airplane model thoroughly, but we were able to test some aspects of it. Figure 6.17 shows a shaded version of the synthetic airplane (a twin engine Cessna) and the normalized mesh we used for processing. The original airplane mesh had large variations in edge length, and the normalization program had a lot of difficulty normalizing the mesh. There seemed to be a lot of unseen connectivity in some of the mesh edges; it was likely not a strictly simple mesh. The processing stages of the Cessna are shown in Figure 6.18. The wings were labeled correctly but the horizontal and vertical stabilizers were not found, because during working set generation they were marked as all one region. Naturally none of the constraints for the stabilizers could be tested; there were not enough constructed features left to even hypothesize a

Table 6.1: Distance calculations for coffee cups. Mean Distance is the average distance from the labeled constructed feature to the manually marked points.

Cup 1 - Mean Distance (mm)

	5K	10K	20K
Rim	4.15	3.16	2.83
Bottom Edge	3.87	2.29	2.04
Bottom Surf.	0.39	0.37	0.56
Handle	4.78	4.62	4.14

Cup 2 - Mean Distance (mm)

	5K	10K	20K
Rim	3.53	3.38	2.79
Bottom Edge	2.14	2.23	2.04
Bottom Surf.	2.45	1.57	0.67
Handle	6.02	5.02	4.28

Cup 3 - Mean Distance (mm)

	5K	10K	20K
Rim	4.34	2.93	2.77
Bottom Edge	7.29	4.37	2.33
Bottom Surf.	1.67	0.16	0.18
Handle	7.99	6.41	5.32

Cup 4 -Mean Distance (mm)

	5K	10K	20K
Rim	1.92	2.48	1.57
Bottom Edge	2.98	2.11	1.29
Bottom Surf.	0.57	0.71	0.53
Handle	/	6.43	4.11

Cup 5 - Mean Distance (mm)

	5K	10K	20K
Rim	5.65	9.99	2.86
Bottom Edge	18.3	3.32	10.3
Bottom Surf.	2.71	0.49	0.29
Handle	7.16	8.64	4.51

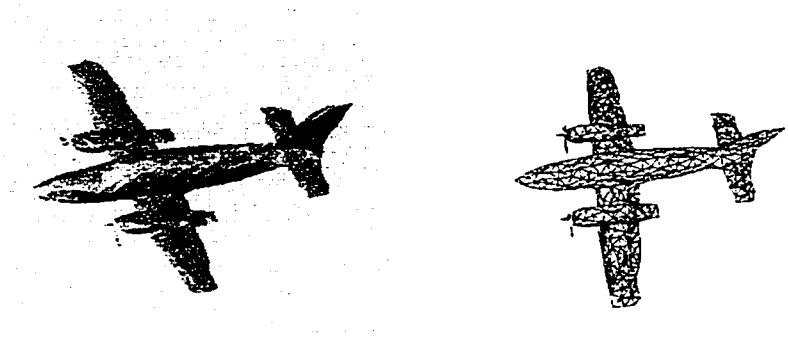


Figure 6.17: Shaded rendering and mesh representation of a synthetic airplane.

labeling.

The first real instance is a plastic model of a small Piper. A color image, the resulting mesh, and the results of processing are shown in Figure 6.19. The mesh generation software had difficulty processing this type of object, and there are concavities on the edges of the wings. This caused separate convex working sets to be generated along the one wing. The other wing did not appear to have a clean separation between the wing and the body, and the wing region grew into the region around the body. Changing the curvature threshold can prevent the merging of the wing with the body edge, but that caused more separate regions along the wing edge. No features were identified in this example.

The third airplane we tried to generate was a Beechcraft. The mesh generation was even worse than the Piper, and we include the results here only for completeness. The results are shown in Figure 6.20. Many constructed sets were formed, but no features were labeled as being wings or stabilizers.

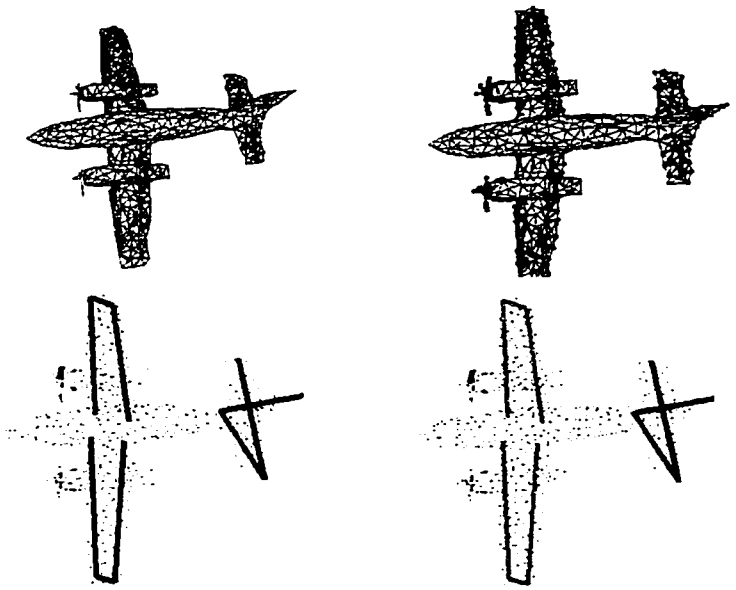


Figure 6.18: Processing stages for synthetic airplane. The working sets are shown as red-convex, cyan-concave, green-planar, and yellow-other. The line group constructed features are shown in blue. The labeled features are shown in orange. Only the wing edges were detected.

6.2.5 Pipe Fittings

We used five different pipe fittings to explore the feasibility of a simple object recognition system based on our symbolic model. Figure 6.21 shows an intensity image of each pipe fitting along with the mesh generated by our system. Figures 6.22 through 6.35 show the processing stages for all of the pipe fittings. Part (a) is the normalized mesh. Part (b) shows the working sets obtained from the bottom up stage of processing. Constructed features are shown in part (c). Since we were only interested in the relationships between the rims of the fittings, only circular constructed features were generated with the convex working sets. Part (d) shows the final feature labeling. Both circles of a parallel pair are shown in the same color, while circles of a perpendicular pair are shown in contrasting colors.

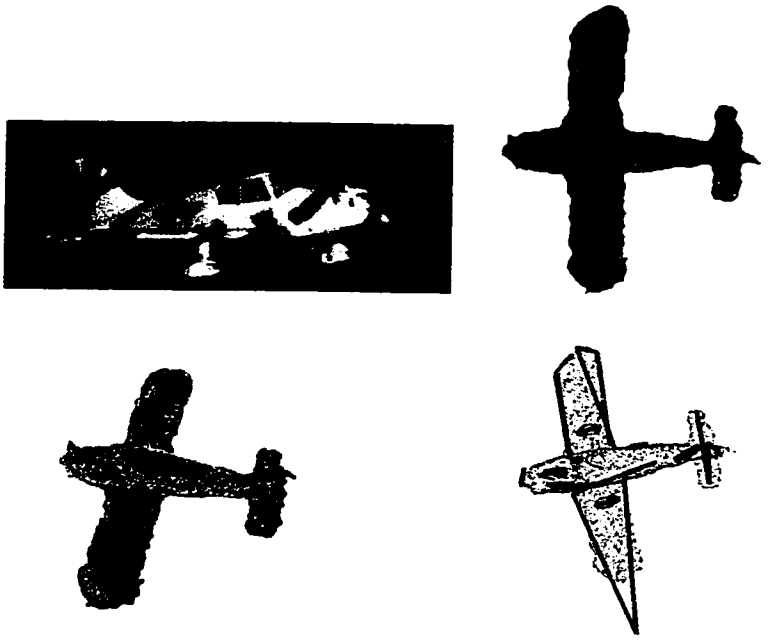


Figure 6.19: Processing stages for Piper airplane. The working sets are shown as red-convex, cyan-concave, green-planar, and yellow-other. The line group constructed features are shown in blue. No features were labeled during the feature labeling stage of processing.

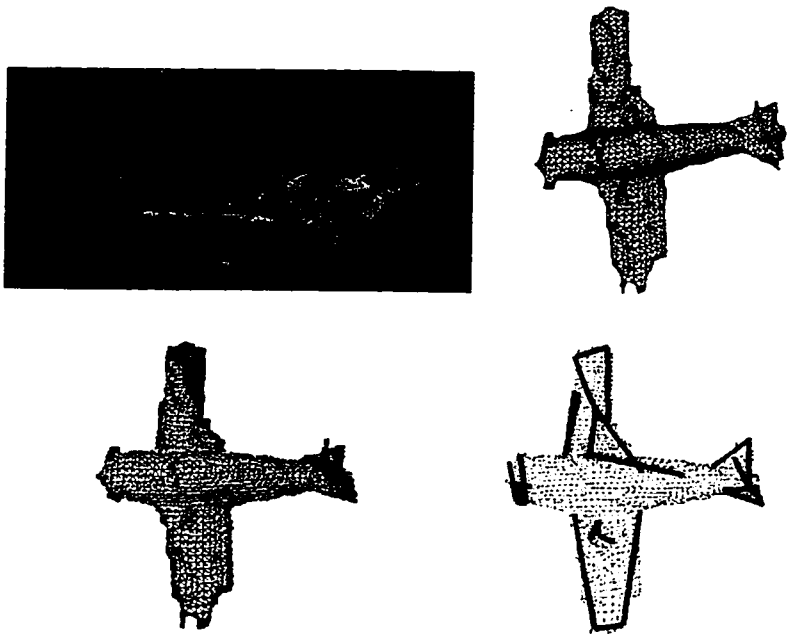


Figure 6.20: Processing stages for Beechcraft airplane. The working sets are shown as red-convex, cyan-concave, green-planar, and yellow-other. The line group constructed features are shown in blue. No features were labeled during the feature labeling stage of processing.

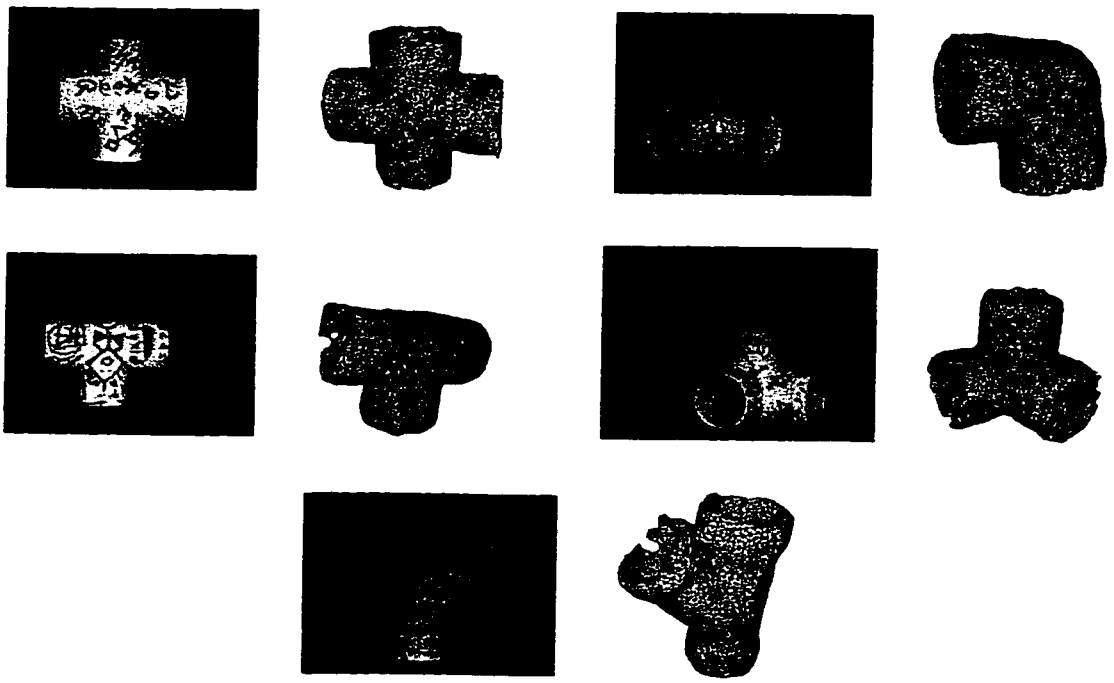


Figure 6.21: The five instances of "pipe fitting" used in experiments. On the left is the intensity image of one view of each fitting, while on the right is a mesh constructed by our system after normalization and reduction to approximately 20K triangles.

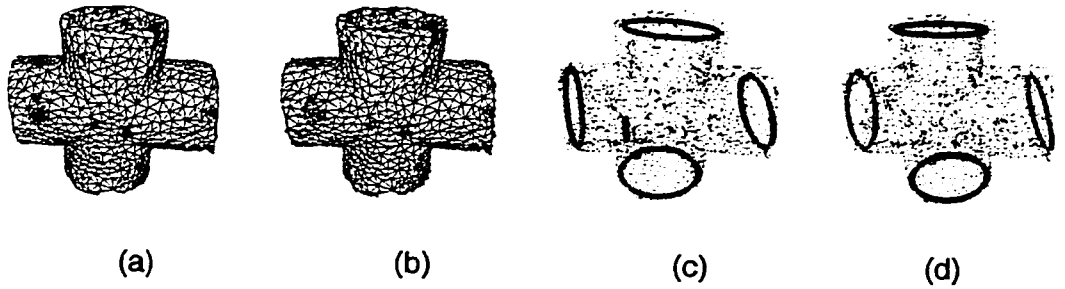
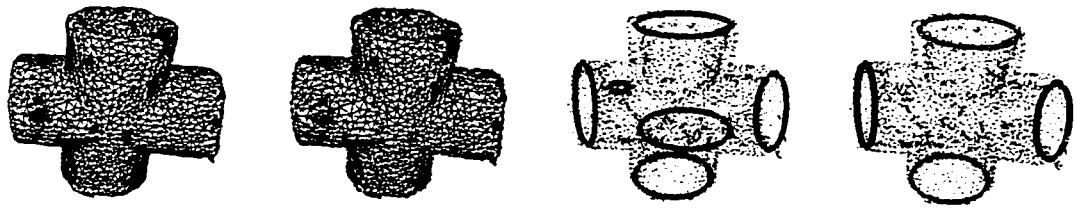
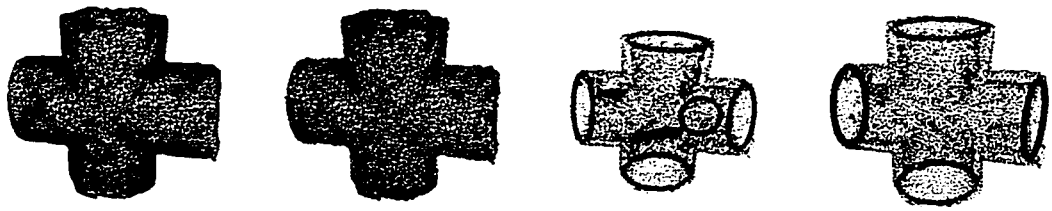


Figure 6.22: Processing stages for cross pipe fitting with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.



(a) (b) (c) (d)

Figure 6.23: Processing stages for cross pipe fitting with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.



(a) (b) (c) (d)

Figure 6.24: Processing stages for cross pipe fitting with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.

Figure 6.25: Processing stages for elbow pipe fitting with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. In this case the object could not be identified, so no found features are displayed.

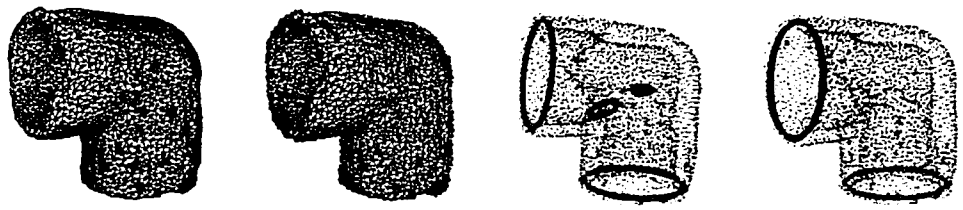


Figure 6.26: Processing stages for elbow pipe fitting with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.

Figure 6.27: Processing stages for elbow pipe fitting with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.

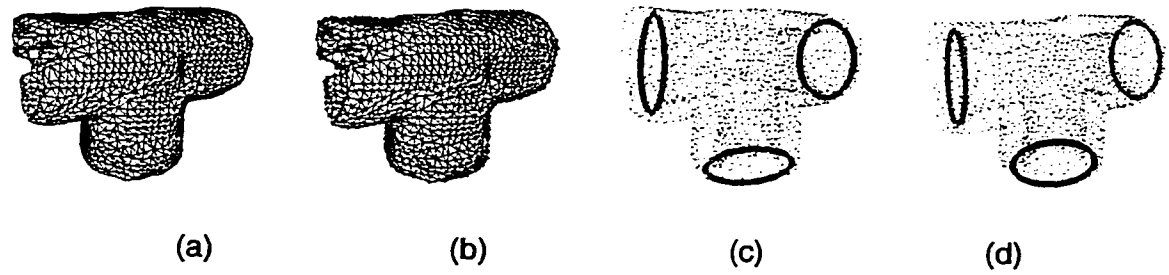
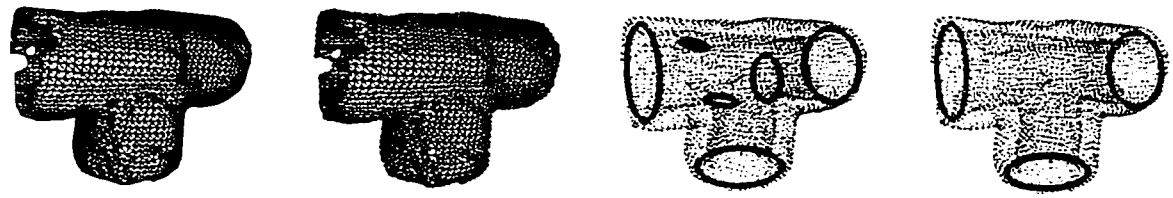
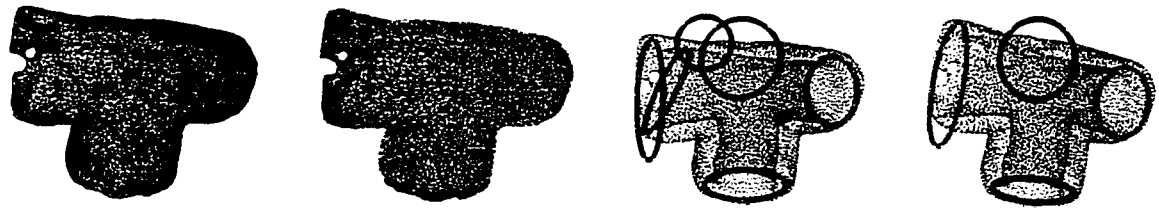


Figure 6.28: Processing stages for tee pipe fitting with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.



(a) (b) (c) (d)

Figure 6.29: Processing stages for tee pipe fitting with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.



(a) (b) (c) (d)

Figure 6.30: Processing stages for tee pipe fitting with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d) with both circles of a parallel pair shown in the same color and circles of a perpendicular pair shown in contrasting colors.

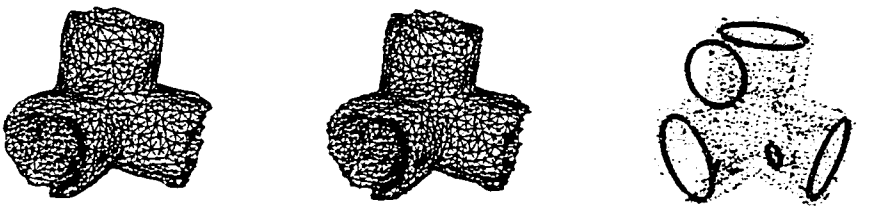


Figure 6.31: Processing stages for tricorner pipe fitting with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. In this case the object could not be identified, so no found features are displayed.

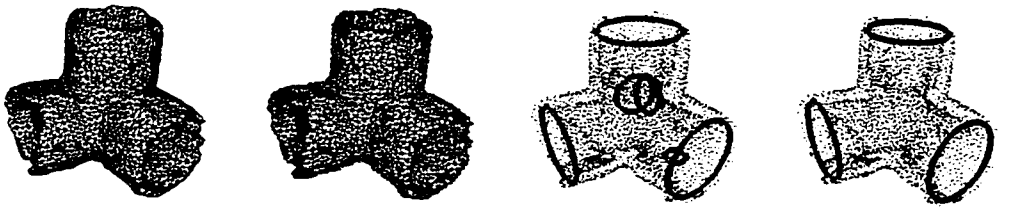
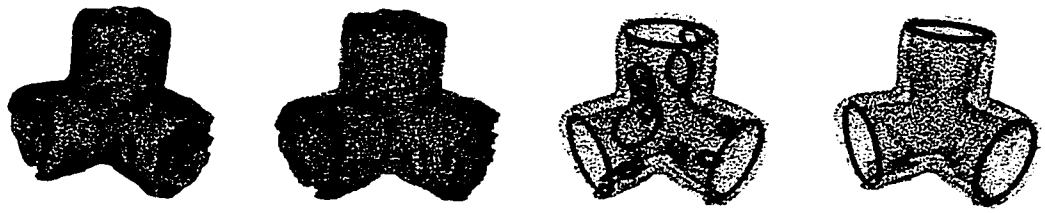
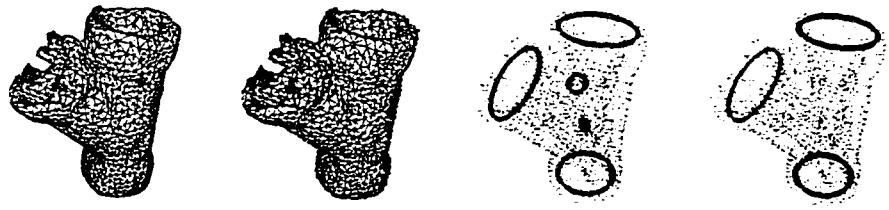


Figure 6.32: Processing stages for tricorner pipe fitting with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d).



(a) (b) (c) (d)

Figure 6.33: Processing stages for tricorner pipe fitting with a mesh resolution (a) of 20K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d).



(a) (b) (c) (d)

Figure 6.34: Processing stages for wye pipe fitting with a mesh resolution (a) of 5K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d).

(a) (b) (c) (d)

Figure 6.35: Processing stages for wye pipe fitting with a mesh resolution (a) of 10K faces. The working sets are shown in (b) as red-convex, cyan-concave, green-planar, and yellow-other. In (c), circular constructed features are generated using the convex working sets, and are shown in red. Found features are shown in (d).

The relationship table and relationship definitions from chapter 5 are repeated in Table 6.2 for ease of reference. Table 6.3 is a list of the pipe fittings used and which relationships were determined to be satisfied, along with the identification of the fitting.

Most pipe fittings were classified properly. The two that were classified as unknown were due to an additional, unexpected constraint being met by a constructed feature that was very similar to the constructed features for the real rims. One tee was classified properly, but the correct perpendicular rim could not be distinguished from a constructed feature caused by noise. Finally, the wye fittings were classified properly, but in one case a circle fit to noise was detected as one of the rims instead of the actual rim.

Table 6.2: Relationships satisfied by pipe fitting type. Bullet in a column indicates that the relationship must be satisfied.

Relationships Satisfied by Pipe Fitting Type						
	Relationship					
		r_1	r_2	r_3	r_4	r_5
Type of Pipe Fitting	wye	•		•		
	cross	•			•	•
	elbow		•			
	tricorner		•		•	
	tee	•			•	

Relationships $R = \{r_1, r_2, r_3, r_4, r_5\}$

$r_1 = (rim1, rim2, concentric)$

$r_2 = (rim1, rim2, perpendicular)$

$r_3 = (rim1, rim3, cdist(0, \frac{bb(O):a}{2}))$

$r_4 = (rim1, rim3, perpendicular)$

$r_5 = (rim3, rim4, concentric)$

Table 6.3: Relationships satisfied by pipe fitting meshes. Any unsatisfied or extra satisfied relationship causes misclassification of object type.

Relationships Satisfied by Experimental Meshes						
	r_1	r_2	r_3	r_4	r_5	*Classification*
cross 5K	•			•	•	cross
cross 10K	•			•	•	cross
cross 20K	•			•	•	cross
elbow 5K		•	•			unknown
elbow 10K		•				elbow
elbow 20K		•				elbow
tee 5K	•			•		tee
tee 10K	•			•		tee
tee 20K	•			•		tee
tricorner 5K		•	•	•		unknown
tricorner 10K		•		•		tricorner
tricorner 20K		•		•		tricorner
wye 5K	•		•			wye
wye 5K	•		•			wye

Chapter 7

SUMMARY AND FUTURE WORK

7.1 Summary

In this dissertation we have defined a 3D symbolic spatial model to represent generic object classes. We have presented a new methodology for detecting predefined features in object meshes, and a method for linking the symbolic model with a real instance of an object mesh. We have developed and implemented a prototype system using the model and methodology.

7.2 Weaknesses of Implementation

We have shown the abstract model and the theory to be feasible approaches to the problem of recognizing and/or labeling the features of abstract object classes, but the current implementation has some weaknesses.

- Any error in detection of working sets cannot be overcome in later stages of processing. Some systems allow error in recognition to guide resegmentation and further analysis. This type of iterative process could be beneficial.

- The labeling algorithm used here expects a fully consistent labeling. Approaches that allow error up to a threshold, find the "best" or least-error mapping, or perform a probabilistic labeling should give better results in the final stage of matching.

- In the pipes experiments, unsatisfied or extra satisfied constraints caused misclassification of the object. Linking the working model to the abstract model and reasoning about the connectivity of the edges and surfaces may help make this use of the methodology more robust.

7.3 Future Work

There are many avenues to explore in the future work using this methodology. First, it would be interesting to link the abstract model to the working model and attempt full segmentation and identification of the points on a mesh corresponding to the features in the abstract model. Also, modification of the methodology for use with partial meshes would be useful. Partial range meshes are more common than full ones. Since the implementation is independent of the methodology, it would be interesting to implement a surface or volume extraction technique in the working model.

BIBLIOGRAPHY

[1] F. Arman and J. K. Aggarwal, "CAD-Based Object Recognition in Range Images Using Pre-compiled Strategy Trees",*Three-dimensional Object Recognition Systems*, A. Jain and P. Flynn (eds.), Elsevier Science Publishers BV, 1992, pp. 115-134.

[2] R. Bajcsy and F. Solina. "Three Dimensional Object Representation Revisited", *Proc. 1st Intl. Conf. on Computer Vision*, London, UK, June 87, pp. 231-240.

[3] D. Ballard and C. Brown, Computer Vision, Prentice Hall, New Jersey, 1982.

[4] N.M. Barnes and Z.Q. Liu. "A Knowledge Based Approach to Shape from Shading", in *Proceedings, IEEE International Conference on Intelligent Signal Processing*, Beijing, October 97, pp 1390-1395.

[5] A. H. Barr. "Superquadrics and Angle-preserving Transformations," *IEEE Computer Graphics Appl. 1*, Vol 1, Jan 1981, pp. 11-23.

[6] R. Bergevin and M. D. Levine. "Generic Object Recognition: Building and Matching Coarse 3-D Descriptions from Line Drawings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, January 1993, pp. 19-36.

[7] P. J. Besl. *Surfaces in Range Image Understanding*, Springer-Verlag, New York, 1988.

[8] P. J. Besl. "Triangles as a Primary Representation", *Object Representation in Computer Vision: Proceedings, International NSF-ARPA Workshop*, New York City, NY, December 94, pp 191-206.

[9] I. Biederman. "Human Image Understanding: Recent Research and a Theory," *Computer Vision, Graphics and Image Processing*, Vol. 32, 1985, pp. 29-73.

[10] T. O. Binford. "Visual Perception by Computer," *IEEE Conference on Systems and Control*, Miami, FL, December 1971.

[11] R. A. Brooks. "Symbolic Reasoning Among 3-D Models and 2-D Images," *Artificial Intelligence*, Vol. 17, Nos. 1-3, 1981, pp. 285-348.

[12] C. S. Chua and R. Jarvis, "Point Signatures: A New Representation for 3-D Object Recognition," *International Journal of Computer Vision*. Vol. 25, No. 1, 1997, pp. 63-85.

[13] B. Curless and M. Levoy, "Better Optical Triangulation through Space-time Analysis," *Proceedings, Fifth International Conference on Computer Vision*, pp. 987-994, June 1995.

[14] L. DeFloriani. "Feature Extraction from Boundary Models of Three-Dimensional Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 8, August 1989, pp. 785-798.

[15] S. J. Dickinson, D. Metaxas and A. Pentland. "The Role of Model-Based Segmentation in the Recovery of Volumetric Parts from Range Data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 3, March 1997, pp. 259-267.

[16] S. J. Dickinson, R. Bergevin, I. Biederman, J. O. Eklundh, R. Munck-Fairwood, A. K. Jain, and A. Pentland. "Panel Report: The Potential of Geons for Generic 3-D Object Recognition," *Image and Vision Computing*, Vol. 15, No. 4, April 1997, pp. 277-92.

[17] D.Dion, Jr., D. Laurendeau, and R. Bergevin, "Generalized Cylinders Extraction in a Range Image," in *Proceedings, International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, Ottawa, Ont., Canada, 12-15 May 1997, pp 141-147.

[18] C. Dorai and A. K. Jain. "COSMOS-A Representation Scheme for 3-D Free-Form Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 10, October 1997, pp. 1115-1130.

[19] C. Dorai and A. K. Jain. "Shape Spectrum Based View Grouping and Matching of 3-D Free-Form Objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 10, October 1997, pp. 1139-1145.

[20] C. Dorai and A. K. Jain. "Recognition of 3-D Free-Form Objects," *Proc. of the 13th Intl. Conf. on Pattern Recognition*, Vienna, Austria, 25-29 August 1996, pp. 697-701.

[21] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle. "Multiresolution Analysis of Arbitrary Meshes," *Technical Report No. 95-01-02* Department of Computer Science and Engineering, University of Washington, January, 1995.

[22] T. Fan, G. Medioni, and R. Nevatia. "Recognizing 3-D Objects Using Surface Descriptions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 11, November 1989, pp. 1140-1157.

[23] O. D. Faugeras and M. Herbert. "The Representation, Recognition, and Locating of 3-D Objects," *The International Journal of Robotics Research*, Vol. 5, No. 3, Fall 1986, pp. 27-52.

[24] O. Faugeraus. Three-Dimensional Computer Vision, MIT Press, Cambridge, Massachusetts, 1993.

[25] F. P. Ferrie, J. Lagarde and P. Whaite. "Darboux Frames, Snakes, and Superquadrics: Geometry from the Bottom Up," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 8, August 1993, pp. 771-783.

[26] P. J. Flynn and A. K. Jain. "On Reliable Curvature Estimation," Proceedings CVPR '89, Washington, DC, 1989, pp 110-116.

[27] R. B. Fisher, A. W. Fitzgibbon and D. Eggert. "Extracting Surface Patches from Complete Range Descriptions", *Proceedings, Intl. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, Ottawa, Ont., Canada, 12-15 May 1997 pp. 148-54.

[28] J. Foley, A. van Dam, S. Feiner and J. Hughes. Computer Graphics: Principals and Practice, Addison-Wesley, New York, 1990.

[29] A. D. Gross and T. E. Boult. "Error of Fit Measures for Recovering Parametric Solids", *Proc. 2nd Intl. Conf. on Computer Vision*, Tarpon Springs, FL, December 1988, pp. 690-694

[30] A. Gupta and R. Bajcsy. "Recognition of Superquadric Models in Dense Range Data,", *Three-dimensional Object Recognition Systems*, A. Jain and P. Flynn (eds.), Elsevier Science Publishers BV, 1992, pp. 285-310.

[31] R. Haralick and L. Shapiro. Computer and Robot Vision, Addison-Wesley, New York, 1993.

[32] R.M. Haralick, and L.G. Shapiro, " The Consistent Labeling Problem - Part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. PAMI-1, No. 2, 1979, pp. 173-184.

[33] D. Hernandez. Qualitative Representation of Spatial Knowledge, Springer-Verlag Berlin Heidelberg, Germany, 1994.

[34] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon and R. B. Fisher. "An Experimental Comparison of Range Image Segmentation Algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, No. 7, July 1996, pp. 673-688.

[35] Z. Hongbin, T. Hoshide and T. Hasegawa. "Superquadric-based Object Modeling by and Iterative Segmentation-and Recovery Algoritm," *Proceedings of the SPIE*,Vol. 3208, 1997, pp. 518-29.

[36] D. J. Itter and A. K. Jain. "3-D Surface Discrimination from Local Curvature Measures," *Proceedings CVPR '85: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*,IEEE Comput. Soc. Press, Silver Spring, MD, 1985, pp 119-23.

[37] J. L. Mundy. "The Image Understanding Environment Program," *IEEE Expert*, Vol. 10, No. 6, December 1995, pp. 64-73.

[38] A. E. Johnson and M. Herbert. "Control of Polygonal Mesh Resolution for 3-D Computer Vision," *Graphical Models and Image Processing*, Vol. 60, No. 4, July 1998, pp. 261-285.

[39] A. E. Johnson and M. Herbert. "Recognizing Objects by Matching Oriented Points," *Proc. 1997 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, San Jaun, PR, 17-19 June 1997, pp. 684-689.

[40] A. E. Johnson and M. Herbert. "Efficient Multiple Model Recognition in Cluttered 3-D Scenes," *Proc. 1997 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, Santa Barbara, CA, 23-25 June 1998, pp. 671-677.

[41] A. Johnson, P. Leger, R. Hoffman, M. Hebert, J. Osborn. "3-D Object Modeling and Recognition for Telerobotic Manipulation" *Proc. IEEE Intel. Robots and Systems* Vol. 1, August 1995, pp. 103-110.

[42] M. Kass and D. Terzopoulos. "SNAKES: Active Contour Models," *International Journal of Computer Vision*, Vol. 1, 1988, pp. 321-332.

[43] A. Lejeune and F. P. Ferrie. "Partitioning Range Images Using Curvature and Scale", *Proc. 1993 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition*, New York, NY, June 1993, pp. 800-801.

[44] A. Leonardis, A. Jaklic and F. Solina. "Superquadrics for Segmenting and Modeling Range Data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 11, November 1997, pp. 1289-95.

[45] P. J. Neal, L. G. Shapiro and C. Rosse. "The Digital Anatomist Structural Abstraction: a Scheme for the Spatial Description of Anatomical Entities", *Proceedings 1998 American Medical Informatics Association Annual Symposium*, Lake Buena Vista, FL, 7-11 November 1998.

[46] M. Okutomi and T. Kanade. "A Multiple-baseline Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 15 no. 4, pp. 353-363, April 1993.

[47] A. P. Pentland. "Perceptual Organization and the Representation of Natural Form", *Artificial Intelligence*, Vol. 28, May 1986, pp. 293-331.

[48] J. Ponce. "Straight Homogeneous Generalized Cylinders: Differential Geometry and Uniqueness Results," *International Journal of Computer Vision*, Vol. 4, 1990, pp. 79-100.

[49] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. Numerical Recipes in C, Cambridge University Press, New York, 1992.

[50] K. Pulli. "Surface Modeling and Display from Range and Color Data", *PhD. Thesis*, University of Washington, Department of Computer Science and Engineering, 1997.

[51] N. S. Raja and A. K. Jain. "Recognizing Geons from Superquadrics Fitted to Range Data," *Image and Vision Computing*, Vol 10, 1992, pp. 179-190.

[52] N. S. Raja and A. K. Jain. "Obtaining Generic Parts from Range Images Using Multi-view Representation", *CVGIP: Image Understanding,* Vol. 60, No. 1, July 1994, pp. 44-64.

[53] C. Rosse, L. G. Shapiro and J. F. Brinkley. "The Digital Anatomist Foundational Model: Principals for Defining and Structuring its Concept Domain," *Proceedings 1998 American Medical Informatics Association Annual Symposium*, Lake Buena Vista, FL, 7-11 November 1998.

[54] K. Sengupta and K. Boyer. "Organizing Large Structural Modelbases," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 4, pp. 321-332, April 1995.

[55] F. Stein and G. Medioni. "Structural Indexing: Efficient 3-D Object Recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, 1992, pp. 125-145.

[56] F. Solina, A. Leonardis and A. Macerl. "A Direct Part-level Segmentation of Range Images Using Volumetric Models, " *Proc. IEEE Intl. Conf. on Robotics and Automation*, San Diego, CA, May 1994, pp. 2254-9.

[57] L. Stark and K. W. Bowyer. *Generic Object Recognition Using Form and Function*. New Jersey. World Scientific Publishing, 1996.

[58] G. Stockman, J. Chen, Y. Cui. "Measuring Body Points on Automobile Drivers using Multiple Cameras," Computer Science Dept., Michigan State University. July 5, 1996.

[59] H. Tanahashi, N. Murakami, K. Yamamoto. "Recovery of Superquadric Primitives from a Range Image Using GA," *Proceedings of the SPIE*, Vol. 2910, 1997, pp. 28-33.

[60] G. Taubin, "Discrete surface signal processing: the polygon as the surface element," *Object Representation in Computer Vision: Proceedings, International NSF-ARPA Workshop*, New York City, NY, December 94, pp 167-175.

[61] R. Tsai. "Multiframe image point matching and 3-D surface reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, pp. 159-173, March 1993.

[62] R. Tsai, "A Versatile Camera Calibration Technique for High Accuracy 3-D Machine Vision Metrology Using Off-the-shelf Cameras and Lenses," *IEEE Journal of Robotics and Automation,* vol. 3 no. 4, pp. 323-344, August 1987.

[63] E. R. Van Dop, P. P. L. Regtien, "Fitting Undeformed Superquadrics to Range Data: Improving Model Recovery and Classification," *Proc. 1998 IEEE Computer Society Conf. On Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998, pp. 396-401.

VITA

Pamela Neal earned a Bachelor of Science degree in Electrical Engineering from Utah State University in 1986. She returned to academics in 1990, earning a M.Sc. in Electrical Engineering from the University of Washington in 1992. She taught Electrical Engineering at the U.S. Air Force Academy from 1992-1996, and returned to the University of Washington in 1996 to pursue a Ph.D. under an Air Force-sponsored faculty preparation program. She received her Ph.D. in 1999 and returned to the Department of Electrical Engineering at the Air Force Academy as an assistant professor.