Spin-Images: A Representation for 3-D Surface Matching

Andrew Edie Johnson

CMU-RI-TR-97-47

August 13th, 1997

Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in the field of Robotics

Thesis Committee:

Martial Hebert, Chair
Katsushi Ikeuchi
Paul Heckbert
Richard Szeliski, Microsoft Research

This research was funded in part by the Department of Energy through the Federal Energy Technology Center under contract DE-AC21-92MC29104 and in part by the National Science Foundation under Grant IRI-9224521.

Keywords: shape representation, 3-D surface matching, object recognition, spin-images, surface mesh, surface registration, object modeling, scene clutter.

Dedicated to Dorothy D. Funnell, a believer in higher education.

Abstract

Surface matching is the process that compares surfaces and decides whether they are similar. In three-dimensional (3-D) computer vision, surface matching plays a prominent role. Surface matching can be used for object recognition; by comparing two surfaces, an association between a known object and sensed data is established. By computing the 3-D transformation that aligns two surfaces, surface matching can also be used for surface registration.

Surface matching is difficult because the coordinate system in which to compare two surfaces is undefined. The typical approach to surface matching is to transform the surfaces being compared into representations where comparison of surfaces is straightforward. Surface matching is further complicated by characteristics of sensed data, including clutter, occlusion and sensor noise.

This thesis describes a data level representation of surfaces used for surface matching. In our representation, surface shape is described by a dense collection of oriented points, 3-D points with surface normal. Using a single point basis constructed from an oriented point, the position of other points on the surface can be described by two parameters. The accumulation of these parameters for many points on the surface results in an image at each oriented point. These images, localized descriptions of the global shape of the surface, are invariant to rigid transformations. Through correlation of images, point correspondences between two surfaces can be established. When two surfaces have many point correspondences, they match. Taken together, the oriented points and associated images make up our surface representation. Because the image generation process can be visualized as a sheet spinning about the normal of a point, the images in our representation are called *spin-images*.

Spin-images combine the descriptive nature of global object properties with the robustness to partial views and clutter of local shape descriptions. Through adjustment of spin-image generation parameters, spin-images can be smoothly transformed from global to local representations. Since they are object-centered representations, spin-images can be compared without alignment of surfaces. However, spin-images are constructed with respect to specific surface points, so they can also be used to align surfaces. Because spin-images are constructed without surface fitting or optimization, they are simple to construct and analyze.

We demonstrate the usefulness of spin-images by applying them to two problems in 3-D computer vision. First we show that surface registration using spin-images is accurate enough to build complete models of objects from multiple range images. Without a calibrated image acquisition system, we have built twenty models of objects with complicated shapes. We also apply spin-images to the problem of recognizing complete 3-D models of objects in partial scenes containing clutter and occlusion. Using spin-images, we have simultaneously recognized multiple objects from a library containing twenty models. We also verify experimentally that spin-image matching is robust to scene clutter by recognizing objects in 100 scenes containing clutter and occlusion.

.

Acknowledgments

There are many people who helped me obtain my Ph.D. First and foremost, I would like to thank my thesis advisor, Martial Hebert. Martial guided me through this work, and with his constant expressions of “this shouldn’t work,” kept me on track. Martial is smart, funny and easy going; excellent qualifications for any advisor.

I developed most of this thesis while working on the Artisan Project. Jim Osborn, the project manager for Artisan, kept me funded and drove me to apply my research to the solution of real problems. This helped me focus my work and demonstrate its merit to the world. I would also like to thank the rest of the Artisan project members including Regis Hoffman, Chris Leger, Sachin Chheda, Kaushik Merchant, Cecelia Sheppard, and Sam Gershon who provided the technical support needed to create the Artisan System.

When I came to CMU, I was basically a math and physics guy. Now, I write software and do little else. From entry to exit I had to ask a lot of questions, and who better to ask than other students? My officemate, David Simon, was an excellent source of all kinds of information. In particular his knowledge of 3-D computer vision and computer graphics were invaluable. My other officemate, Julio Rosenblatt, was always a good source for C++ tricks and parties. Tom Nichols talked to me about statistics and encouraged me to exercise. Besides being the best man in my wedding, John Beacom talked with me about the trials and tribulations of being a graduate student. Harry Shum, Yoichi Sato, Sanjiv Singh, Mark Wheeler, Dongmei Zhang, Pete Rander and Fabio Cozman were friends in the Vision and Autonomous Systems Center who helped me out with discussions and advice. Without the help of Marie Elm, who read and corrected all of my papers, this document would have one quarter the commas, and no semi-colons.

A turning point in my graduate student career came when I spent the summer at Digital Equipment Corporation’s Cambridge Research Lab. While there, Sing Bing Kang, my advisor at CRL, showed me a different perspective on computer vision. This, along with many interesting discussion with my officemate, Tina Kapur, expanded my horizons and re-energized my desire to do research. Shortly after working at CRL, I came up with the idea of the spin-image and proposed my thesis topic six months later.

During my thesis work, my committee members provided needed advice on many topics, and their comments on this document greatly improved its final form. In particular, Paul Heckbert aided development of my mesh resampling algorithm, Katsushi Ikeuchi gave insight into object representations, and Richard Szeliski’s provided expertise in statistics and modeling.

Finally, I would like to thank my family. My parents, Michael and Nancy Johnson have always been a source of constant support and encouragement. From them I have gotten a love of knowledge and the desire to work hard to achieve one’s goals. My grandmother, Dorothy Funnell, always believed in higher education. Her generous support allowed me to spend more time studying and less time earning a living while an undergraduate. My last and final thanks go to my wife, Aimee Noel. She has made my life happy, exciting and fun, and without her I would have had many more stressful and worrisome moments.

Table of Contents

List of Figures

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 6

Chapter 7

Chapter 8

Chapter 9

Chapter 10

Appendix A

Appendix C

List of Tables

Chapter 1
Introduction

Surface matching is a process that determines when the shapes of two different objects are equivalent. This thesis focuses on the problem of automating surface matching through the use of computers. In order to match surfaces, 3-D data measuring the shape of the surface is collected and then transformed into a representation that is suitable for automatic surface matching. The main contribution of this thesis is the development and application of a novel and effective 3-D surface matching representation.

Surface matching is a technique from 3-D computer vision that has many applications in the area of robotics and automation. Through surface matching, an object can be recognized in a scene by comparing a sensed surface to an object surface stored in memory. When the object surface is matched to the scene surface, an association is made between something known (the object) and something unknown (the scene); information about the world is obtained. Another application of surface matching is the alignment of two surfaces represented in different coordinate systems. By aligning the surfaces, the transformation between the surface coordinate systems is determined. Surface alignment has numerous applications including localization for robot navigation and modeling of complex scenes from multiple views. This thesis explores the use of our surface matching representation in solving both the object recognition and surface alignment problems.

In traditional computer vision, the shape or presence of objects in a scene is determined by analyzing camera images, 2-D projections of the 3-D world which contain no explicit information about the depth of objects in the scene. However, recent advances in 3-D sensing technology and shape recovery algorithms have made digitized 3-D surface data widely available. For ex-

ample, contact probes can be touched to the surface of objects to measure individual 3-D points. Laser range scanners can actively scan the surface of the object, returning a dense grid of 3-D surface points. Stereo vision systems can passively determine the 3-D position of features or textured areas from two camera views of an object. Computed Tomography or Magnetic Resonance Imaging can sense 3-D surfaces that are extracted though volumetric image processing. Given that 3-D data can be readily acquired, the next step in surface matching is to transform the data into a form that is appropriate for surface matching.

Shape representations are used to collate the information stored in sensed points so that surfaces can be compared efficiently. Shape can be represented in many different ways, and finding an appropriate representation for shape that is amenable to surface matching is still an open research issue in computer vision [37]. The variation among shape representations spans many different axes. For instance, shape representations can be classified by the number of parameters used to describe each primitive in the representation. Representing objects using planar surface patches [51] uses many primitives, each with a few parameters. On the other hand, representing an object with generalized cylinders [20] requires fewer primitives, but each has many parameters. Another axis of comparison is the local versus global nature of the representation. Deformable surfaces [18] are global representations useful for describing single objects, while surface curvature [14] measures local surface properties and can be used for surface matching in complex scenes. The multitude of proposed surface representations indicates the lack of consensus on the best representation for surface matching.

Another difficulty in choosing an appropriate surface representation is choosing the coordinate system in which to describe the data. Surfaces can be defined in viewer-centered coordinate systems or object-centered coordinate systems. Viewer-centered representations describe surface data with respect to a coordinate system dependent on the view of the surface. Although viewer-centered coordinate systems are easy to construct, the description of the surface changes as viewpoint changes. Therefore, surfaces must be aligned before they can be compared, forcing the need for an alignment procedure. Furthermore, to represent a surface from multiple views, a separate representation must be stored for each different viewpoint.

An object-centered coordinate system describes an object surface in a coordinate system fixed to the object. In object-centered coordinates, the description of the surface is view-independent,

so surfaces can be directly compared, without first aligning the surfaces. Object-centered representations are also more compact than viewer-centered representations because a single surface representation describes all views of the surface. Unfortunately, finding an object-centered coordinate system is difficult because these systems are generally based on global properties of the surface. However, if an object-centered coordinate system can be extracted robustly from surface data, then it should be used over a viewer-centered coordinate system.

In 3-D object recognition, an important application of surface matching, an object surface is looked for in a scene surface. If an object-centered surface matching representation is to be used to recognize objects in real scenes, it must be robust to clutter and occlusion. Real scenes contain multiple objects, so surface data sensed in the real world will contain clutter, surfaces that are not part of the object surface being matched. Because clutter will corrupt global properties of the scene data, generating object-centered coordinate systems in the cluttered scenes is difficult. The usual method for dealing with clutter is to segment the scene into object and non-object components [1][51]; naturally, this is difficult if the position of the object is unknown. An alternative to segmentation is to construct object-centered coordinate systems using local features detected in the scene [33][59]; here again there is the problem of differentiating object features from non-object features. Another difficulty occurs because surface data often has missing components i.e., occlusions. Occlusions will alter global properties of the surfaces and therefore, will hinder construction of object-centered coordinate systems.

This thesis presents a novel object-centered shape representation that can effectively match surfaces even in the presence of clutter and occlusion. In addition, our representation is general: it places few restrictions on object shape and topology; it is applicable to many different problems solvable by surface matching; and it is sensor independent.

1.1 A Representation for Surface Matching

In our representation, surface shape is described by a dense collection of 3-D points and surface normals. In addition, associated with each surface point is a descriptive image that encodes global properties of the surface in an object-centered coordinate system. By matching images, correspondences between surface points can be established resulting in surface matching. Taken together, the points, normals and associated images make up our surface representation.

Figure 1-1 shows the components of our surface matching representation.

Representing surfaces using a dense collection of points is feasible because many 3-D sensors and sensing algorithms return a dense sampling of surface shape. Furthermore, from sensor geometry and scanning patterns, the adjacency on the surface of sensed 3-D points can be established. Using adjacency and position of sensed 3-D points surface normal can be computed. We use a polygonal surface mesh to combine information about the position of 3-D surface points and the adjacency of points on the surface. In a surface mesh, the vertices of the surface mesh correspond to 3-D surface points, and the edges between vertices convey adjacency. Given enough points, any object can be represented by points sensed on the object surface, so surface meshes can represent objects of general shape. Surface meshes can be generated from different types of sensors and do not generally contain sensor-specific information; they are sensor-independent representations. The use of surface mesh as representations for 3-D shapes has been avoided in the past due to computational concerns. However, processing power has reached a level where computations using surface meshes are now feasible [8][89].

Our approach to surface matching is based on matching individual surface points in order to match complete surfaces. Two surfaces are said to be similar when many points from the surfaces are similar. By matching points, we are breaking the problem of surface matching into

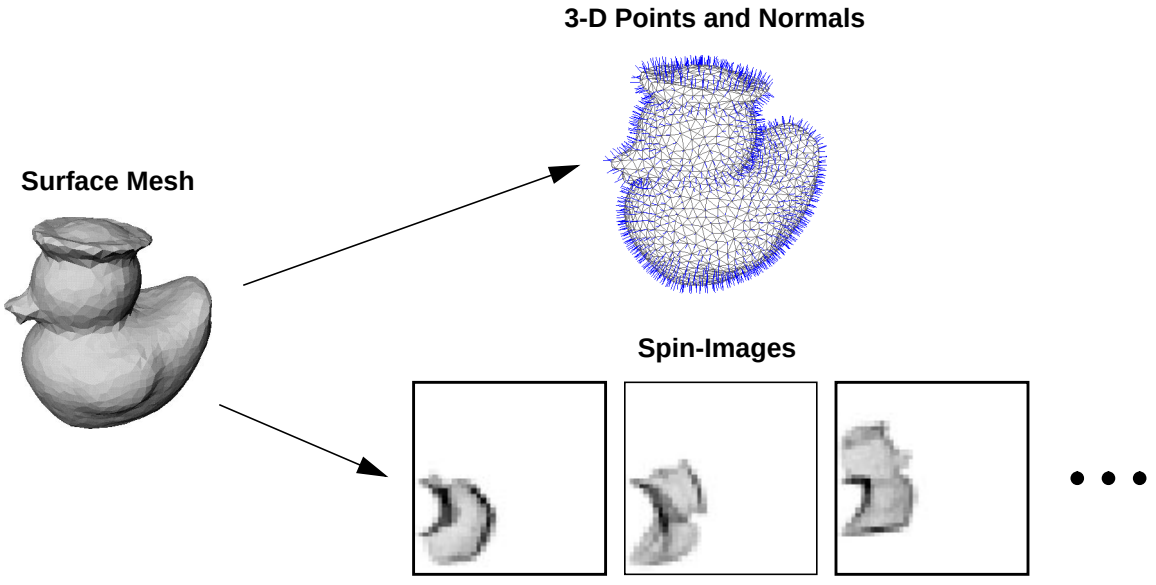


Figure 1-1: Components of our surface representation. A surface described by a polygonal surface mesh can be represented for matching as a set of 3-D points, surface normals and spin-images.

many smaller problems. Consequently, matching points provides a method for handling clutter and occlusion in surface matching; clutter points on one surface will not have matching points on the other, and occluded points on one surface will not be searched for on the other. If many points between the two surfaces match then the surfaces can be matched. The main difficulty with matching surfaces in this way is describing surface points so that they can be differentiated from one another, while still allowing point matching in scenes containing clutter and occlusion.

The idea of matching points to match surfaces is not a novel concept. Stein and Medioni [86] recognize 3-D objects by matching points using structural indexing and their “splash” representation. Similarly, Chua and Jarvis [14] match points to align surfaces using principal curvatures. Our methods differs from these in the way that points are represented for matching and the way that points, once matched, are grouped to match surfaces.

To differentiate among points, we construct 2-D images associated with each point. By comparing images, points can be matched. Specifically, these images are created by constructing a local basis at an oriented point (3-D point with surface normal) on the surface of an object. As in geometric hashing [59][60], the positions with respect to the basis of other points on the surface of the object can then be described by two parameters. By accumulating these parameters in a 2-D array, a descriptive image associated with the oriented point is created. Because the image encodes the coordinates of points on the surface of an object with respect to the local basis, it is a local description of the global shape of the object and is invariant to rigid transformations. Since 3-D points are described by images, we can apply powerful techniques from 2-D template matching and pattern classification to the problem of surface matching.

To distinguish our point matching representation from camera images common in computer vision we have chosen the name *spin-image*; *image* because the representation is a 2-D array of values, and *spin* because the image generation process can be visualized as a sheet spinning about the normal of the point. Spin-images combines the descriptive nature of global object properties with the robustness to partial views and clutter of local shape descriptions. Through adjustment of spin-image generation parameters, spin-images can be transformed smoothly from global to local representations. Since they are object-centered representations, spin-images can be compared without alignment of surfaces. Furthermore, spin-images are constructed

with respect to specific surface points, so they can be used to align surfaces when desired.

Spin-images are simply transformations of the surface data; they are created by projecting 3-D points into 2-D images. Spin-images do not impose a parametric representation on the data, so they are able to represent surfaces of general shape. Spin-images are constructed without surface fitting or optimization, so they are simple to construct, analyze and modify to meet specific needs.

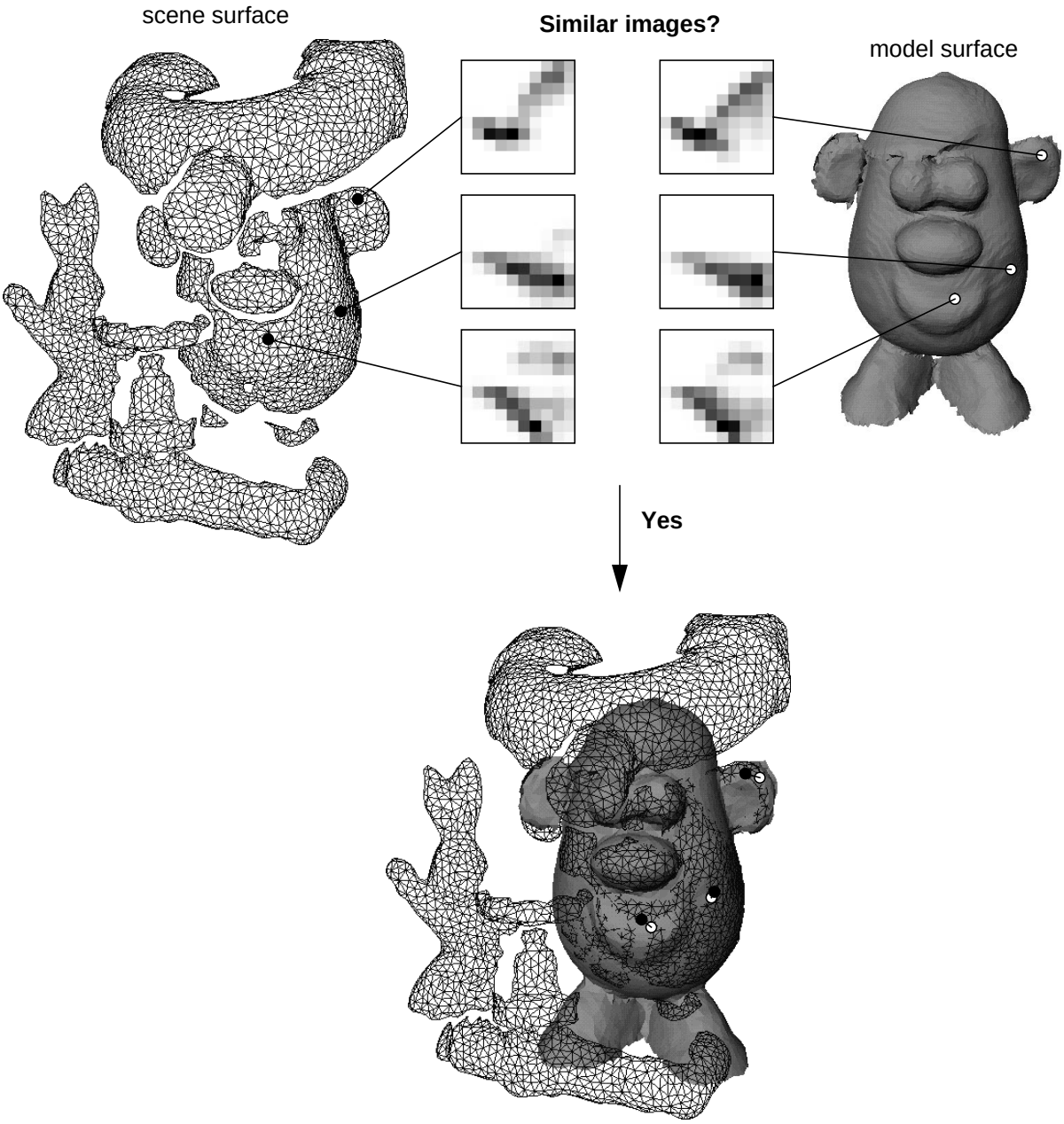


Figure 1-2: Surface matching concept

Spin-images generated for two corresponding points on different surfaces will be similar, so oriented points can be matched based on a comparison of their spin-images. Spin-images are descriptive enough that correspondences between points can be established based on a comparison of images alone. Since spin-images can be generated for any point on the surface of an object, our algorithm does not require or use feature extraction. Furthermore, through localiza-

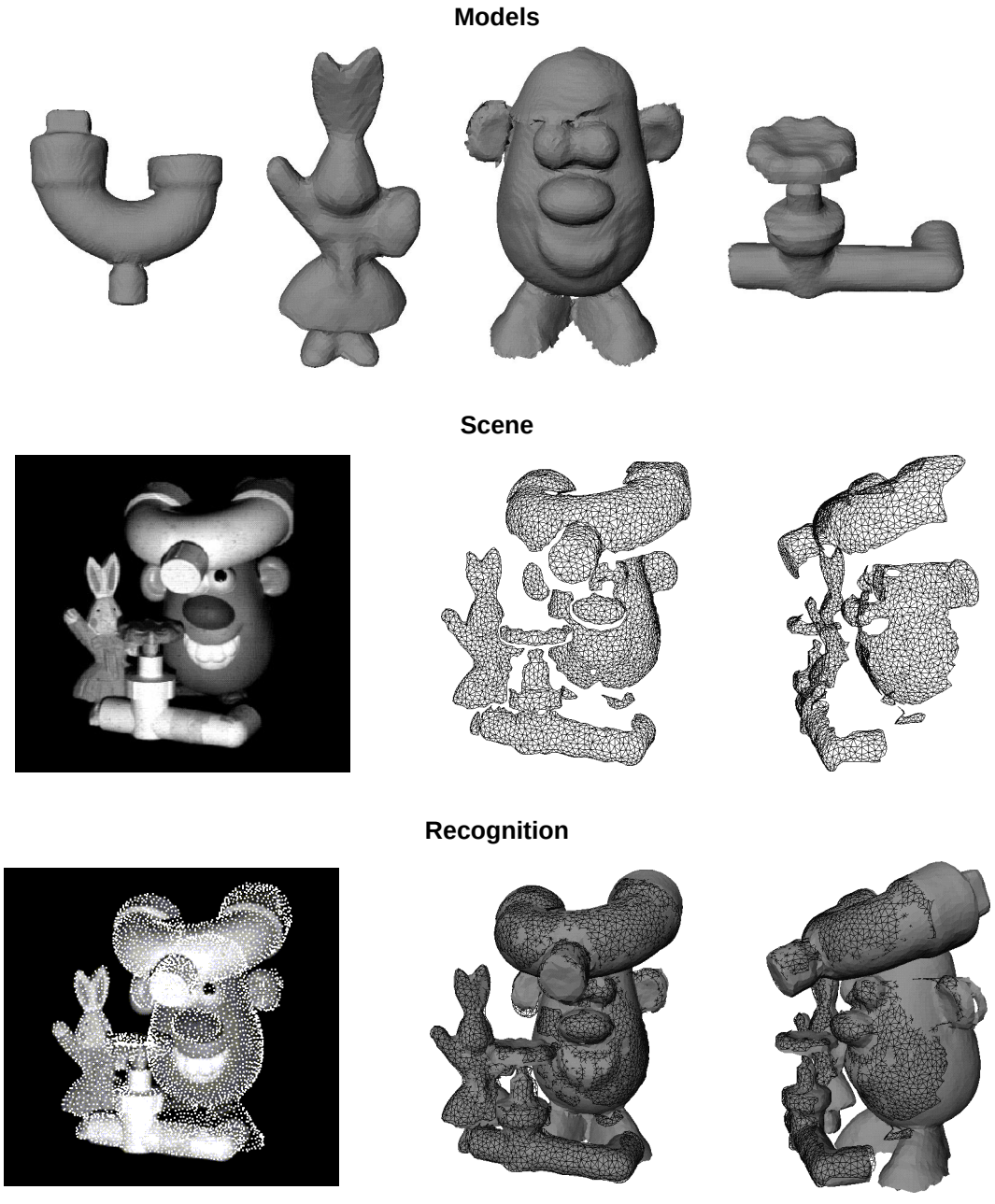


Figure 1-3: An example of object recognition with spin-images. Complete object models (top) are searched for in the scene data (middle). Through spin-image matching, the models are found in the scene and aligned with the scene surfaces (bottom).

tion of spin-image support, our representation can be made robust to clutter and occlusion, so segmentation of scene data is not necessary for surface matching in cluttered scenes. Figure 1-2

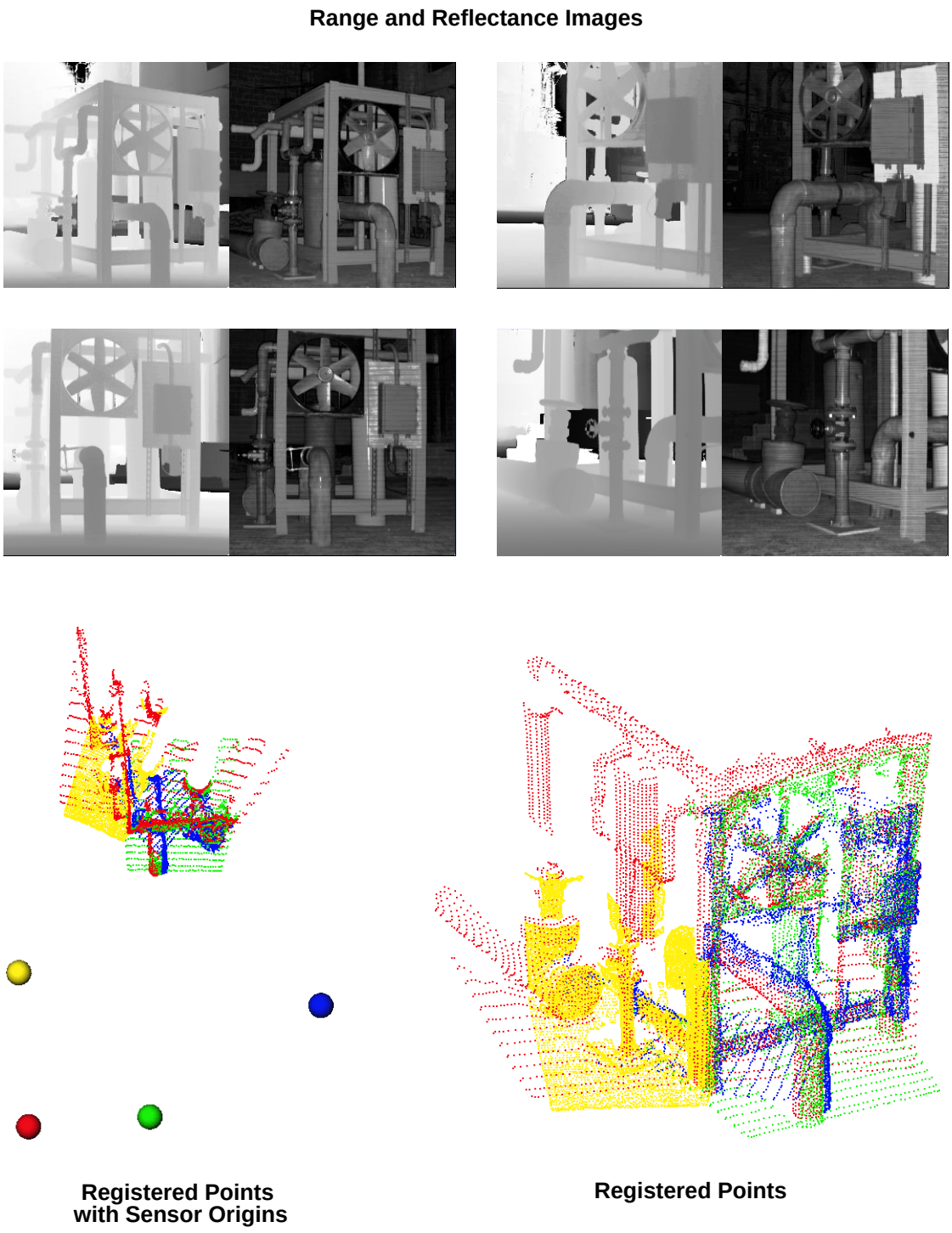


Figure 1-4: An example of surface registration using spin-images. Four range images are taken of an industrial facility (top). The surfaces are registered by matching spin-images even when the sensor viewpoints vary drastically (bottom).

shows the basic concept behind surface matching. Spin-images are constructed for points on the two surfaces being compared. If many pairs of images are similar, then the surfaces are said to match. To verify that the surfaces actually match, point matches established by matching spin-images can be used to align the surfaces. Alignment places the surfaces in the same coordinate system, so direct distance comparisons can occur.

Surface matching using spin-images has two main applications: object recognition and surface alignment. An example where spin-images are used for object recognition is shown in Figure 1-3. Here, complete object models are searched for in a partial scene. The proximity of the objects creates a scene with large amounts of clutter and occlusion. In spite of this, using spin-images, all of the models in the scene are simultaneously recognized and localized in their correct scene positions. This thesis presents results showing the simultaneous recognition of multiple objects in cluttered scenes from libraries containing up to twenty models. It also presents an experimental analysis of recognition rate versus the clutter and occlusion using one hundred different scenes. This analysis shows that surface matching using spin-images degrades gracefully, not catastrophically, with the addition of clutter and occlusion to a scene, making spin-images an effective representation for object recognition in complex scenes.

An example demonstrating the use of spin-images for surface alignment is shown in Figure 1-4. Here, four range images of an industrial structure taken from widely different viewpoints are properly aligned. Using spin-images for surface alignment allows the alignment of surfaces when the transformation between views is completely unknown. To demonstrate the applicability of surface alignment, this thesis shows how spin-images can be used for to build complete models of single objects from multiple unregistered range images. By building models of twenty objects, we show that spin-images can be used to align a wide variety of shapes.

In summary, our surface matching representation has the following advantages:

- matches general shapes
- robust to clutter and occlusion
- applicable to a wide variety of computer vision problems
- sensor independent
- simple to construct and analyze

1.2 Thesis Overview

This thesis presents a new representation for 3-D surface matching called the spin-image. In addition to describing the creation and matching of spin-images, this thesis applies spin-image matching to two common problems in computer vision: surface alignment and object recognition. In the first two chapters we describe how spin-images are created and matched in order to align surfaces. Since object modeling from multiple unregistered range images is a direct application of surface alignment, we explain this application next. We then describe the use of spin-images in object recognition and give results showing recognition of tens of objects in hundreds of scenes. To demonstrate surface matching in real scenes, this thesis also describes a system for semi-automatic 3-D world modeling that uses a spin-image matching engine. When required, analysis of assumptions and parameters in spin-image creation and matching are given throughout. Furthermore, instead of placing related work in a single separate section, this thesis describes related work in its context in the thesis. Consequently, multiple related work sections appear throughout the thesis. The breakdown by chapter is as follows:

In Chapter 2 we describe the how spin-images are created from a polygonal surface mesh. We explain the spin-image generation parameters and show how, by setting parameters during spin-image generation, spin-images can be smoothly transformed from global to local representations. Our similarity measure for comparison of spin-images is also explained in this chapter.

Chapter 3 describes the surface matching engine. The surface matching engine compares spin-images and produces matches between oriented points on the two surfaces being compared. The matching engine then filters these point correspondences and groups them using geometric consistency. Once grouped, point correspondences are used to compute a transformation that aligns the surfaces. The final component of the surface matching engine is a procedure that verifies the alignment of two surfaces. Chapter 3 also gives surface matching results from many different sensing domains and situations showing the versatility of our representation, It concludes with a comparison of spin-images to related representations and matching strategies. The work described in Chapter 2 and Chapter 3 was published in the paper “Surface registration by matching oriented points” [48].

In Chapter 4 we describe a technique for building complete 3-D models of objects from multiple unregistered range views, a direct application of the surface matching engine. Using this technique we have built twenty models of complex objects from multiple range images. In addition to demonstrating that spin-images can be used to match a wide variety of surface shapes, these results are used as models in an object recognition system described in the following chapters. Some of the modeling work presented in this paper was described in “Registration and integration of textured 3-D data” [53].

Chapter 5 explains how surface matching using spin-images can be applied to model-based object recognition. We explain how multiple surfaces are stored in a model library and then simultaneously matched to scene data. We also describe a theoretical model of the effect of clutter on matching spin-images, and give details of this model in Appendix C. The conclusion: matching of spin-images can be made robust to clutter by appropriate choice of spin-image generation parameters. The use of spin-images for object recognition was described in “Object recognition by matching oriented points” [50].

Spin-images are redundant representations of surface shape. Chapter 6 explains how spin-images can be compressed using Principal Component Analysis to speed up spin-image comparisons and storage. By compressing spin-images, object recognition with large model libraries is made possible. In Chapter 6 we introduce our four algorithms for object recognition, each one based on a different form of compression.

In Chapter 7, we show numerous recognition results in complex scenes containing occlusion and clutter. Chapter 7 also details our recognition experiments which characterize the performance of recognition in the presence of clutter and occlusion. The experiments show that matching spin-images is essentially unaffected by clutter while the performance of matching degrades gracefully as the amount of model occlusion increases. Results are also shown for the four algorithms presented in the previous chapter along with analysis of the algorithms as the number of models in the model library increases.

A real world application of object recognition is building 3-D models of complex interiors. Chapter 8 describes a system that interactively builds up a model of a robot workspace using object recognition based on spin-image matching. Since the system described was integrated

with a real robot and demonstrated in a real setting, it demonstrates the matching with spin-images is possible outside of the laboratory. This system was described in the paper “A system for semi-automatically modeling of complex environments” [52].

Spin-images are a simple way to represent 3-D shape, so they lend themselves to analysis. Chapter 9 presents an analysis of the parameters in spin-image generation and details some rules for setting these parameters. It also shows how spin-image matching performance is affected as sensing conditions like noise and surface sampling change.

Chapter 10 outlines future directions of research. Spin-images are a useful way of representing 3-D shape and all of the applications of spin-images have not been exhausted. Chapter 11 presents our conclusions and contributions.

Appendix A describes an important algorithm for controlling the resolution of surfaces described by meshes. This algorithm is used to improve spin-image matching and it has applications in coarse-to-fine recognition of objects. This appendix is practically verbatim from “Control of mesh resolution for 3-D computer vision” [49]. Appendix B gives the mathematical derivation of our spin-image comparison function, and Appendix C describes a theoretical model of the effect of clutter on spin-image matching.

Chapter 2
Spin-Images

We have developed a 3-D surface matching representation that combines the descriptive nature of global object properties with the robustness to partial views and clutter of local shape descriptions. Specifically, a local basis is computed at an oriented point (3-D point with surface normal) on the surface of an object represented as a polygonal surface mesh. The positions with respect to the basis of other points on the surface of the object can then be described by two parameters. By accumulating these parameters in a 2-D array, a descriptive *spin-image* associated with the oriented point is created. Because the spin-image encodes the coordinates of points on the surface of an object with respect to the local basis, it is a local description of the global shape of the object and is invariant to rigid transformations. Since our representation comprises images that describe 3-D points, we can apply powerful techniques from 2-D template matching and pattern classification to the problem of 3-D object recognition.

This chapter explains how spin-images are generated from surfaces represented as polygonal meshes. First, we motivate our use of polygonal mesh for shape description. Next, we describe how spin-images are created from polygonal meshes. We end by describing how different spin-images can be compared and matched. The next chapter explains how spin-images are used for matching surfaces and

2.1 Representing Shape Using Surface Meshes

Our algorithm for surface matching assumes that surfaces are represented as polygonal surface meshes. A polygonal mesh is a piece-wise linear surface that comprises vertices, edges and faces. A vertex is a 3-D point on the surface, edges are the connections between two vertices, and

a face is a closed sequence of edges. In a polygonal surface mesh, an edge can be shared by, at most, two adjacent polygons and a vertex is shared by at least two edges [30]. Our surface matching representation is composed of two parts: the polygonal mesh describing the shape of the surface, and the spin-images created from the surface mesh.

The use of meshes for representing surfaces has been avoided in the past because of concerns about storage and accuracy. Instead, parametric representations, for example generalized cylinders [20] and super-quadrics [72], have been used to reduce the computation and noise. However, surface meshes have some attractive properties which make them quite useful in computer vision. First and foremost, surface meshes can be used to represent the shape of an object to an arbitrary level of detail by increasing the sampling of the surface. Complex multi-part descriptions of objects become unnecessary [1]. A second advantage of surface meshes is that they can be created directly from sensed 3-D data without fitting or approximation. Sensed 3-D points become the vertices of the mesh and connectivity is established by the 3-D scanning geometry. Since surface meshes are piece-wise linear, computation of distances and intersections between surfaces have closed form solutions. A final benefit, displaying surface meshes is trivial using standard 3-D computer graphics viewers, facilitates analysis and display of results.

Recently, computing power has increased to a level where using polygonal surfaces meshes as a representation for 3-D shape has become feasible. Given the benefits of using surface meshes, we decided to use them as our representation for 3-D surfaces. Other researchers have reached the same conclusion and have started using polygonal meshes directly in their research as well [8] [89]. In addition, the use of polygonal meshes in computer vision can draw on the large body of work in computer graphics where polygonal meshes are a standard way of representing shape. Particularly useful to computer vision researchers are algorithms related to surface fitting [2], multi-resolution modeling [24][39] and mesh smoothing [89].

2.1.1 Mesh Resolution

An important concept when discussing meshes is that of *mesh resolution*. The resolution of a polygonal mesh determines the amount of surface detail the mesh contains and is closely related to the number of vertices, edges and faces in the mesh. A coarse resolution mesh will contain

a small number of vertices, while a fine resolution mesh will contain a large number of vertices. One can think of mesh resolution as the distance between connected vertices or, equivalently, the length of edges in the mesh. When the distances between vertices connected by edges in a mesh varies greatly over the surface of the mesh, the resolution of a mesh is not well defined.

Mesh resolution is defined as the median of all edge lengths in a mesh.

We define the resolution of a mesh to be the median edge length over all of the edges in the mesh. When the lengths of edges in the mesh are similar, then the spacing between vertices will be uniform and our metric for mesh resolution will be meaningful. It should be noted that our definition of resolution has an inverted meaning from the definition of resolution used in image processing. As in image processing, mesh resolution increases as samples are added and decreases samples are removed. However, our metric for resolution, edge length, *decreases* as samples are added and *increases* as samples are removed.

To create meshes with uniform spacing between vertices from meshes with an uneven distribution of vertices, we have developed an algorithm for control of mesh resolution. This algorithm is based on mesh simplification algorithms from computer graphics [35][39][44][76][92] and described in detail in Appendix A. In addition to creating an even sampling of an object's surface, the algorithm can also be used to increase or decrease the resolution of the surface mesh representing the object. This opens up the possibility of hierarchical forms of object recognition.

2.2 Oriented Points

A fundamental component of our surface matching representation is an *oriented point*, a three-dimensional point with an associated direction. Oriented points are static versions of oriented particles [84][98]. Oriented points are used to create spin-images. We define an oriented point O at a surface mesh vertex using the 3-D position of the vertex $\boldsymbol{p}$ and surface normal at the vertex $\boldsymbol{n}$. The surface normal at a vertex is computed by fitting a plane to the points connected to the vertex by the surface mesh. Specifically, surface normal at a vertex is the eigenvector with the smallest eigenvalue of the inertia matrix of vertex and the other vertices directly connected to it by the edges of the surface mesh [22].

The above eigenvector computation does not determine the inside/outside direction of the surface normal; for spin-image generation, oriented points should be oriented to the outside of the object surface. If the surface mesh was created from a sensor with a single viewing direction, then the normal direction can be chosen as the one pointing toward the sensor. Otherwise, surface normals of a mesh must be oriented to the outside of the object using the following heuristic. First, a vertex is chosen and the orientation of its normal is spread to the normals of its adjacent vertices. This process is repeated until the normals of all of the vertices are consistently oriented to the inside or outside of the object. Next the orientation (inside/outside) of all of the normals on the surface is determined by calculating the scalar products of the surface normal at each vertex and the vector from the centroid of the object to the vertex. If the majority of scalar products are positive, the normals have been oriented to the outside. Otherwise, the normals have been oriented to the inside, so they are inverted. If the object has multiple connected components, this normal orientation procedure is applied separately to each connected component. To date, we have never encountered an object where this heuristic would not generate outside oriented surface normals, although objects can be constructed where it will fail. Given this method for computing surface normal, an oriented point can be constructed at each vertex of a surface mesh using the position of the vertex and its surface normal.

As shown in Figure 2-1, an oriented point defines a 5 degree of freedom (DOF) basis ***(p**,**n)*** (i.e., local coordinate system) using the tangent plane P through ***p*** oriented perpendicularly to ***n*** and

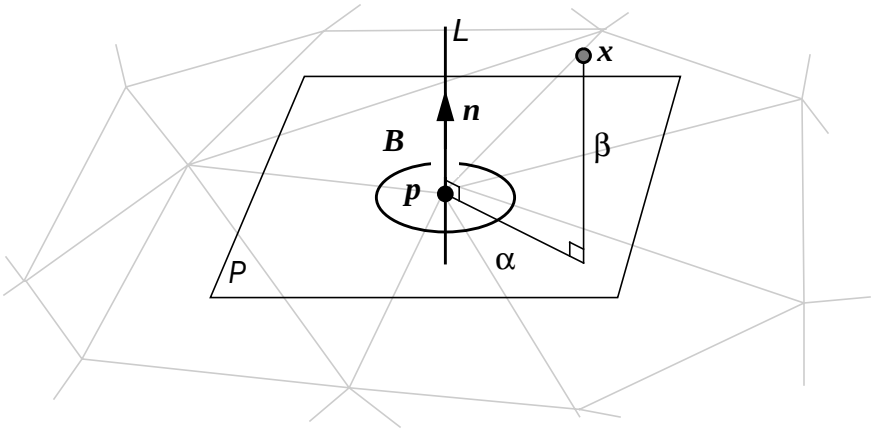


Figure 2-1: An oriented point basis created at a vertex in a surface mesh. The position of the oriented point is the 3-D position of the vertex, and the direction of the oriented point is the surface normal at the vertex. Two coordinates can be calculated given an oriented point: α the radial distance to the surface normal line L and β the axial distance above the tangent plane P.

the line L through $\boldsymbol{p}$ parallel to $\boldsymbol{n}$. The two coordinates of the basis are α, the perpendicular distance to the line L, and β the signed perpendicular distance to the plane P. An oriented point basis is a cylindrical coordinate system that is missing the polar angle coordinate (because this coordinate cannot be determined using just surface position and normal). Using an oriented point basis O, we can define a *spin-map* S_O as the function that projects 3-D points $\boldsymbol{x}$ to the 2-D coordinates of a particular basis $(\boldsymbol{p},\boldsymbol{n})$ corresponding to oriented point O

$$S_O : \boldsymbol{R^3} \rightarrow \boldsymbol{R^2}$$
$$S_O(\boldsymbol{x}) \rightarrow (\alpha, \beta) \;=\; (\sqrt{\|\boldsymbol{x}-\boldsymbol{p}\|^2 - (\boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p}))^2},\ \boldsymbol{n}\cdot(\boldsymbol{x}-\boldsymbol{p})) \qquad (2.1)$$

Although α cannot be negative, β can be both positive and negative.

The term spin-map comes from the cylindrical symmetry of the oriented point basis; the basis can spin about its axis with no effect on the coordinates of points with respect to the basis. A consequence of the cylindrical symmetry is that points that lie on a circle that is parallel to P and centered on L will have the same coordinates (α,β) with respect to the basis.

A well known technique in computer vision is to describe the shape of an object with respect to a coordinate system attached to the object. These *object-centered* coordinate systems have the advantage that they describe the shape of the object independently of its pose. Oriented points are object centered coordinate systems. We use oriented point bases, which determine only two of the three coordinates of points, instead of complete 3-D bases because an oriented point basis can be determined robustly and unambiguously almost everywhere on the surface of an object, while a complete 3-D basis cannot. An oriented point basis is well defined everywhere on the surface of the object except at surface discontinuities, where first order surface derivatives are undefined, so surface normal cannot be computed. When creating a complete three-dimensional coordinate system, three unambiguous axes must be determined. If surface normal, which can be computed reliably, is chosen as one of the axes, then two axes in the tangent plane of the surface must be determined to complete the basis. An obvious choice for these axes would be the directions of principal curvature on the surface [14][28]which, when combined with the surface normal, create the Darboux frame of the surface. However, determination of the directions of principal curvature of a surface require the computation of second-

order surface derivatives, so the resulting axes are more susceptible to noise than surface normal. Furthermore, as the surface approaches a plane, the directions of principal curvature become undefined. Instead of attempting to calculate a stable6 DOF basis at each point, the known stable information (surface normal) is used to compute a 5 DOF basis. Although a spin-map is not a complete coordinate system, it can still be used to describe (albeit incompletely) the position of a point with respect to other points on the surface of an object. In the next section, we will describe how we use this fact to encode the shape of a surface in an object-centered fashion.

2.3 Spin-Image Generation

Each oriented point O on the surface of an object has a unique spin-map S_O associated with it. When S_O is applied to all of the vertices of a surface mesh M, a set of 2-D points is created. In this case, the vertices on the surface of M are the pre-image of the spin-map, and the resulting 2-D points are the image of the spin-map. We will use the term *spin-image* $\mathbf{I}_{O,M}$ to refer to the result of applying the spin-map S_O to the vertices of M. A spin-image is a description of the shape of an object because it is the projection of the relative position of 3-D points that lie on the surface of an object (vertices) to a 2-D space where some of the 3-D metric information is preserved. Figure 2-2 shows three 2-D point sets that result from applying the spin-map at a point on the surface of a rubber duckie mesh model to all of the vertices in the mesh. Figure 2-3 shows three 2-D point sets that result from applying the spin-map at a point on the surface of a valve model to the vertices on the mesh. Since spin-images describe the relative position of points on a rigid object with respect to a particular point on that same object, spin-images are independent of rigid transformations applied to the object and, therefore, are truly object-centered shape descriptions.

Suppose there exists a method for comparison of spin-images. A simple surface matching scheme based on matching spin-images is then possible. Spin-images from vertices in the scene are compared to spin-images from vertices in the model; when a good match occurs, a correspondence between the model vertex and the scene vertex is established. With three or more point correspondences, a rigid transformation between model and scene can be calculated and verified. The problem is then to find an efficient way to compare spin-images.

Variations in viewing direction, sensing modalities and surface mesh construction all combine to prevent the position and connectivity of vertices in the scene and vertices on the model from being the same. Any algorithm that assumes that vertices will have similar positions between model and scene cannot be robust because this assumption cannot be guaranteed. However, the global shape of an object represented by two different surface meshes will be the same. Therefore, when comparing spin-images, the goal is to compare global shape conveyed by the two images while not being distracted by local variations.

To encode global shape while reducing the effects of local variations in vertex position, we originally approached the problem of matching spin-images with a method based on geometric hashing [59]. Our initial algorithm used a hash table to encode geometric constraints between

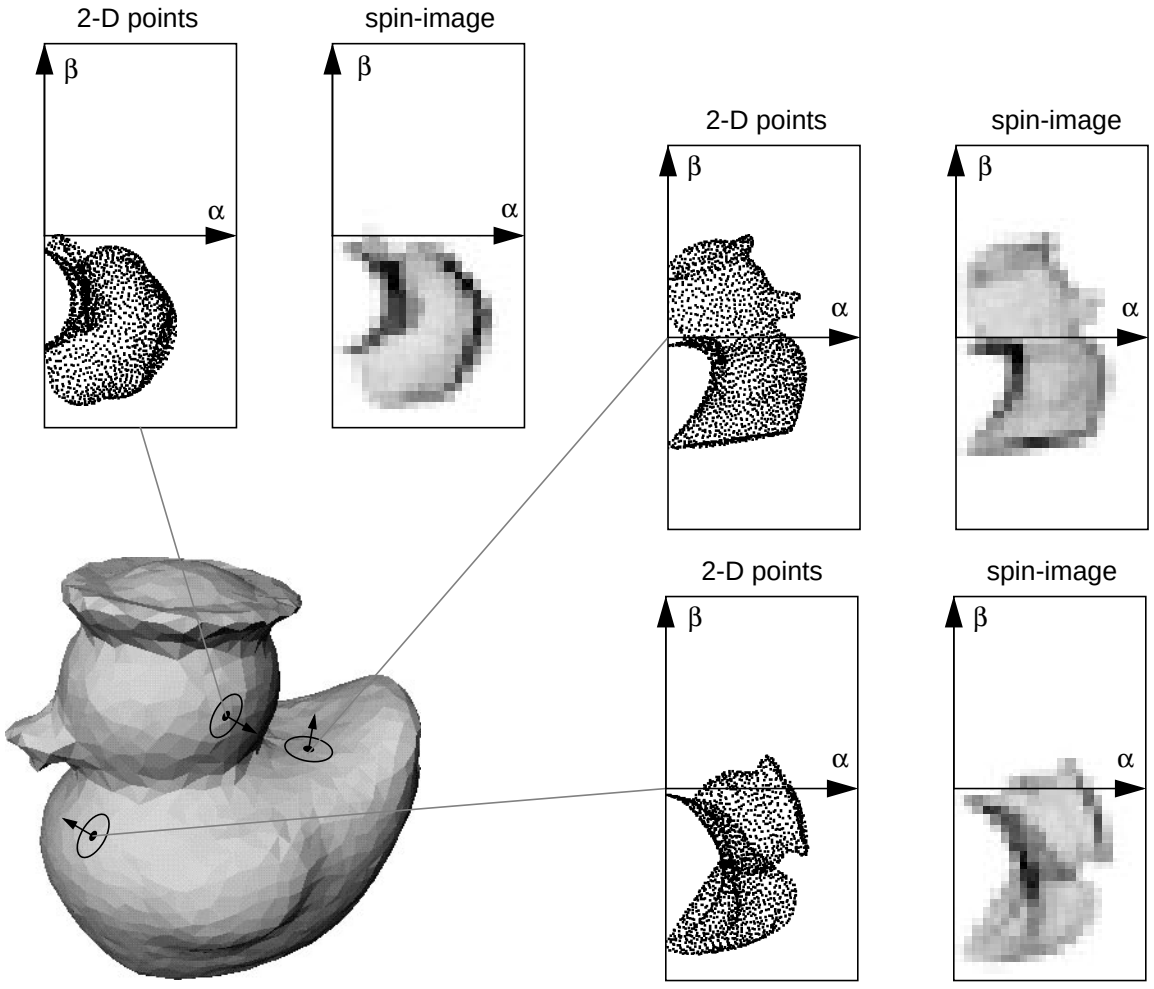


Figure 2-2: Spin-images for three oriented points on the surface of a rubber duckie model. The 3-D position of vertices in the mesh are mapped into 2-D using the spin-map for each oriented point basis. By accumulating 2-D points in discrete bins, spin-images are generated.

points; indices to oriented point bases were stored in the hash table in the bins determined by spin-map coordinates of other points on the object. Point matching proceeded by choosing a point in the scene, computing bin locations from spin-map coordinates of the other points in the scene, and voting for model points with indices in the computed bins. The model point with the highest vote was chosen as the point corresponding to the current scene point. By placing the indices in discrete bins of a hash table, the effect of the exact position of individual points on matching was reduced through averaging.

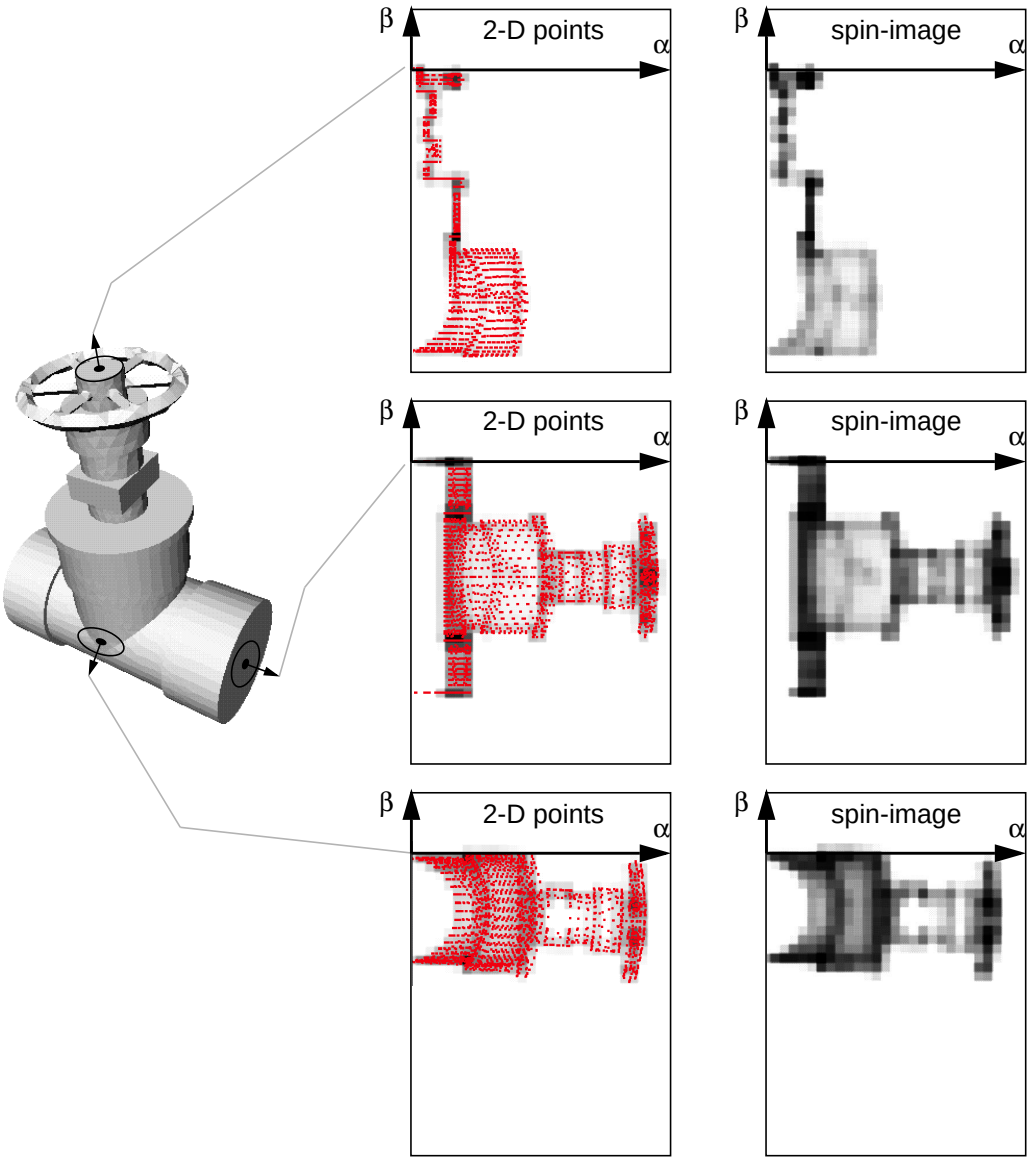


Figure 2-3: Spin-images for three oriented points on the surface of a model of a valve. The 3-D position of vertices in the mesh are mapped into 2-D using the spin-map for each oriented point basis. By accumulating 2-D points in discrete bins, spin-images are generated.

Because all of the points on the surface of the object rather than a small number of feature points are being used, the density of indices in the table will be very high. We discovered that this high density of indices was defeating the purpose of using a hash table. It turns out that a stack of 2-D arrays or accumulators (one for each spin-image) that encode the density of points in each spin-image results in more efficient storage and comparison of spin-images than possible with a hash table. By encoding the density of points in the spin-image, 2-D arrays reduce the effects of local variations in vertex positions while still describing global shape of the object.

To create the 2-D array representation of a spin-image, the procedure described in pseudo-code in Figure 2-5 and pictorially in Figure 2-4 is invoked. First an oriented point O is created from a vertex of the surface mesh. Then for each vertex $\boldsymbol{x}$ on the surface of the object, the spin-map coordinates with respect to O are computed (2.1). If $\boldsymbol{x}$ meets some criteria based on distance from O and angle between O and the surface normal of $\boldsymbol{x}$ (see Section 2.3.2), the bin that its spin-map coordinates index is determined (2.2), and the 2-D array is updated by incrementing the surrounding bins in the array. A simple way to update the array for each point $\boldsymbol{x}$ would be to increment the bin to which the point is spin-mapped by one. However, in order to spread the position of the point in the 2-D array to account for noise in the data, the contribution of the point is bilinearly interpolated to the four surrounding bins in the 2-D array. This bilinear interpolation of the contribution of a point will spread the location of the point in the 2-D array, making the array less sensitive to the position of the point. Once all of the points on the surface of the object have been accumulated, a 2-D array representation of the spin-image is generated.

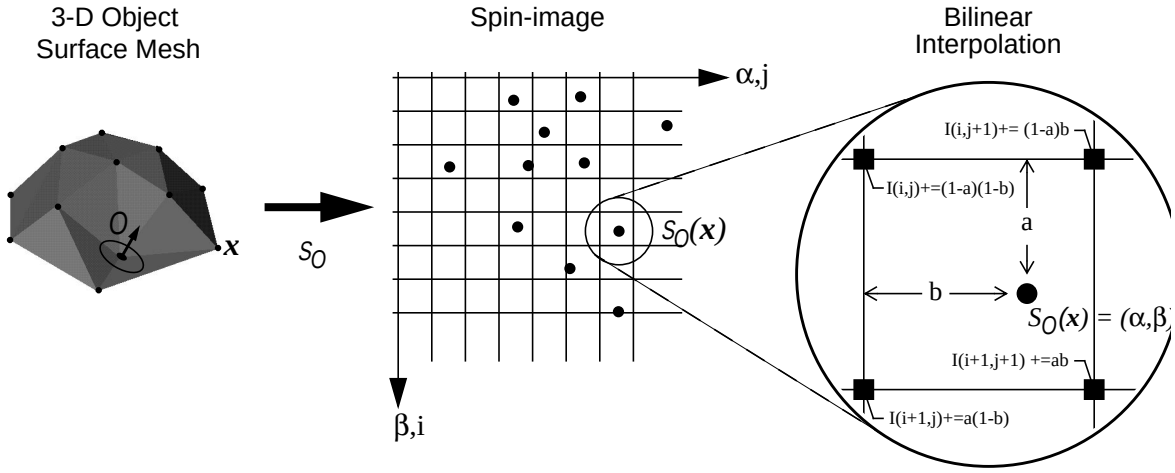


Figure 2-4: The addition of a point to the 2-D array representation of a spin-image.

Figure 2-2 shows the 2-D array representations of three spin-images generated for three oriented points on the surface of a rubber duckie. Figure 2-3 shows some spin images for a model of a valve. The darker the pixel, the higher the number of points projected into that particular bin. Figure 2-6 shows eight animation frames that motivate the name *spin-image*. The spin-image generation process can be visualized as a sheet spinning around the oriented point basis, accumulating space as it sweeps through space. Using this analogy, we can define the sweep volume as the portion of space swept out by the spinning sheet. (It should be noted that sweeping a sheet through space would be an extremely inefficient way to construct spin-images, and consequently, that is not they way spin-images are formed.)

2.3.1 Discussion

Using spin-images to match points opens up the entire field of image-based matching, giving us powerful comparison tools such as image correlation. For the rest of this thesis, we will use the term spin-image to refer to the 2-D array generated from the spin-mapping of points on a 3-D object. Because a spin-image is a global encoding of the surface, it would seem that any disturbance such as clutter and occlusion would prevent matching. In fact, this representation is resistant to clutter and occlusion, assuming that some precautions are taken. Our model of how our representation handles clutter will be described in detail in Chapter 5.

Uniform sampling of surfaces is a requirement for the spin-images of two corresponding points on different instances (surface mesh representations) of the same object to be the similar. This is a much weaker constraint than requiring the positions of points to be the same for the two instances. If the surfaces are uniformly sampled then on average, each corresponding bin of the spin-images will have the same number of points projected into it making the spin-images similar. Figure 2-7 shows two sets of spin-image generated from corresponding vertices on two

```
CreateSpinImage(oriented_point O, spin_image SI, surface_mesh M)
    for all points x on M
        (α,β) = SpinMapCoordinates(O,x)                    // (2.1)
        (i,j) = SpinImageBin(α,β)                          // (2.2)
        (a,b) = BilinearWeights(α,β)                       // (2.3)
        SI(i,j) = SI(i,j) + (1-a)*(1-b)
        SI(i+1,j) = SI(i+1,j) + (a)*(1-b)
        SI(i,j+1) = SI(i,j+1) + (1-a)*b
        SI(i+1,j+1) = SI(i+1,j+1) + a*b
```

Figure 2-5: Pseudo-code for the generation of a 2-D array representation of a spin-image.

different surface model representations of the duckie model. The spin-images generated from corresponding points are similar, even though the coordinates and surface normals of the vertices are not exactly the same. Chapter 9 discussed in detail the effect of surface sampling in spin-image matching.

In cases where surface sampling is not uniform, we use a mesh resampling algorithm (explained in Appendix A) to add and remove points from the surface meshes until the surfaces are uniformly sampled. In most cases, it is not necessary for the surface meshes being matched to have the same resolution as long as they are uniformly sampled. Matching between meshes of different resolutions (a difference of 10 to 1) is demonstrated with elevation maps in Chapter 3 and discussed in detail in Chapter 9.

It is possible that other interpolation schemes could be used to make spin-images more accurate representations of the true shape of an object. For example, a gaussian interpolation kernel could be used instead of a bilinear kernel. Another possibility is to super-sample the faces of the mesh when accumulating points in the spin-images. Both of these interpolation methods would require more processing time than bilinear interpolation; since good performance was obtained with bilinear interpolation, we did not pursue other methods of interpolation.

Spin-images appearance is altered by two types of noise effecting oriented point bases: surface position noise and surface normal noise. Because of averaging during accumulation, surface position noise has little effect on spin-image appearance as long as its magnitude is less than the distance between mesh vertices. On the other hand, the effect of surface normal error on spin-image appearance increases as α and β increase across the spin-image. Consequently, small errors in surface normal can translate into large changes in spin-image appearance; this can result in incorrect oriented point correspondences. However, our surface matching engine eliminates false matches using geometric consistency, so the effect of false matches arising

Figure 2-6: Eight animation frames that motivate the name spin-image. The spin-image generation process can be visualized as a sheet spinning around the oriented point basis, accumulating points as it sweeps space.

from surface normal error can be eliminated from overall surface matching.

Spin-images are rotation and translation invariant, but they are not scale invariant. Consequently, two surfaces of the same shape, but different scale will generate different sets of spin-images. Fortunately, the spin-map is a function of euclidean distances, so the spin-images from scaled versions of the same surface will be the same up to the scale factor between the surfaces. Therefore, multiple models do not have to be stored to represent surfaces at different scales; only the original model and the necessary scale factors need to be stored. Although we do not address the scale issue in this thesis, it does constitute an area of interesting future work.

2.3.2 Spin-Image Generation Parameters

There are three parameters that control spin-image generation: bin size, image width, and support angle. These parameters are defined below along with acceptable ranges of parameter values. More detailed descriptions of the effect of these parameters on spin-image matching are given in Chapter 5 and Chapter 9.

Bin size is a very important parameter in spin-image generation because it determines the storage size of the spin-image and the averaging in spin-images that reduces the effect of individual point positions. It also has an effect on the descriptiveness of the spin-images. The bin size is

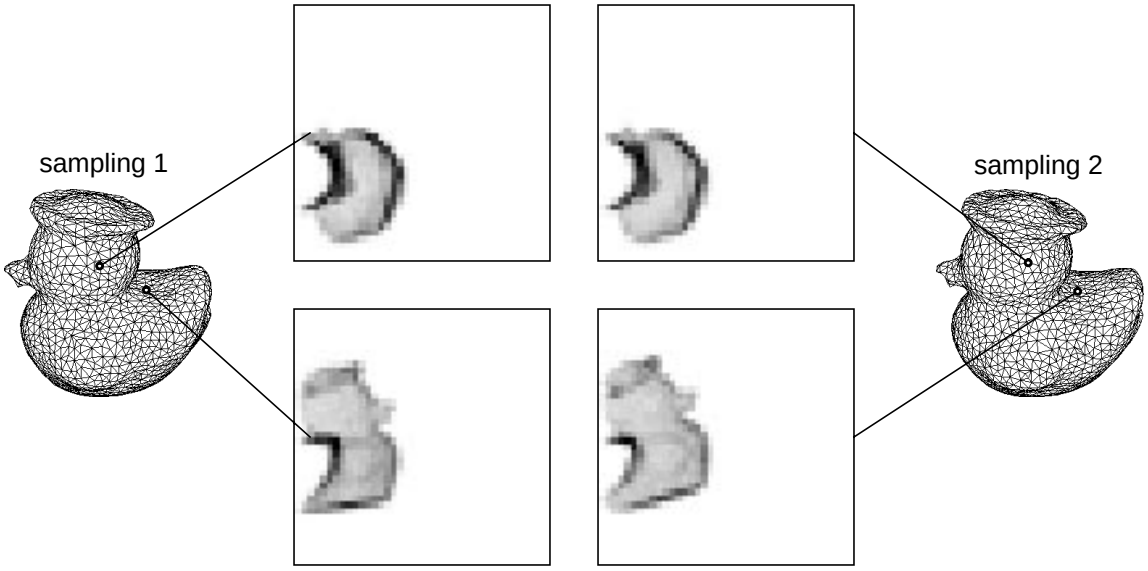


Figure 2-7: Spin-images generated from two different samplings of the duckie model. Although the surface meshes are different, the spin-images generated from corresponding points are similar, even though the coordinates and surface normals of the points are not exactly the same.

set as a multiple of the resolution of the surface mesh (defined in Section 2.1.1) in order to eliminate the dependence of setting bin size on object scale and resolution. Setting bin size based on mesh resolution is feasible because mesh resolution is related to the size of shape features on an object and the density of points in the surface mesh. We have found that setting the bin size to be one to two times the mesh resolution sufficiently blurs the position of individual points in the spin-images while still adequately describing global shape.

Spin-images generated for the duckie model using different bin sizes are shown in Figure 2-8. The spin-image generated for a bin size of four times the model resolution is not very descriptive of the global shape of the model. The spin-image generated with a bin size of one quarter the mesh resolution does not have enough averaging to eliminate the effect of surface sampling. The spin-image generated with a bin size equal to the mesh resolution has the proper balance between encoding global shape and averaging of point positions. A complete description of the effect of bin size on spin-image matching is given in Chapter 9.

Although spin-images can have any number of rows and columns, for simplicity, we generally make the number of rows and columns in a spin-image equal. This results in square spin-images whose size can be described by one parameter. We define the number of rows or columns in a square spin-image to be the *image width*. To create a spin-image, an appropriate image width needs to be determined. Image width times the bin size is called the spin-image *support distance* D_s; support distance determines the amount of space swept out by a spin-image. By setting the image width, the amount of global information in a spin-image can be controlled.

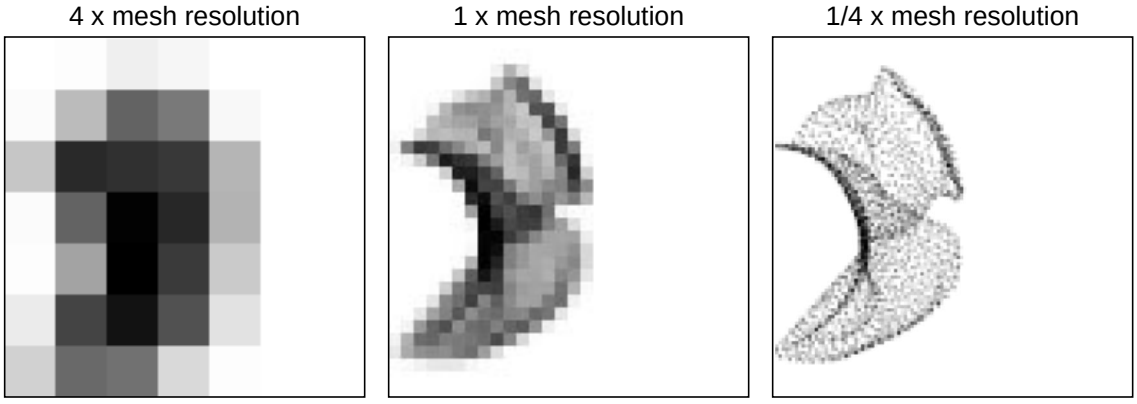


Figure 2-8: The effect of bin size on spin-image appearance. Three spin-images of decreasing bin-size for a point on the duckie model are shown. Setting the bin size to the model resolution creates descriptive spin-images while averaging during point accumulation to eliminate the effect of individual vertex positions.

For a fixed bin-size, decreasing image width will decrease the descriptiveness of a spin-image because the amount of global shape included in the image will be reduced. However, decreasing image width will also reduce the chances of clutter corrupting a spin-image. These trade-offs are discussed at length in the description of the clutter model in Chapter 5 and the effect of image width on spin-image descriptiveness in Chapter 9. Figure 2-9 shows spin-images for a single oriented point on the duckie model as the image width is decreased. In general, we set the image width so that the support distance is on order of the size of the model. If the data is very cluttered, then we set the image width to a smaller value. For the results presented in this thesis, image width is set between 10 and 20 resulting in spin-images with 100 to 400 bins.

Given the bin size b and image width W, the spin-image bin, for particular spin-map coordinates, can be computed. The equations relating spin-map coordinates (α,β) and spin-image bin (i,j) are

$$i = \left\lfloor \frac{\frac{W}{2} - \beta}{b} \right\rfloor \qquad j = \left\lfloor \frac{\alpha}{b} \right\rfloor \tag{2.2}$$

where $\lfloor f \rfloor$ is the floor operator which rounds f down to the nearest integer. Because the distance to the tangent plane of an oriented point can be both positive and negative, the spin-image

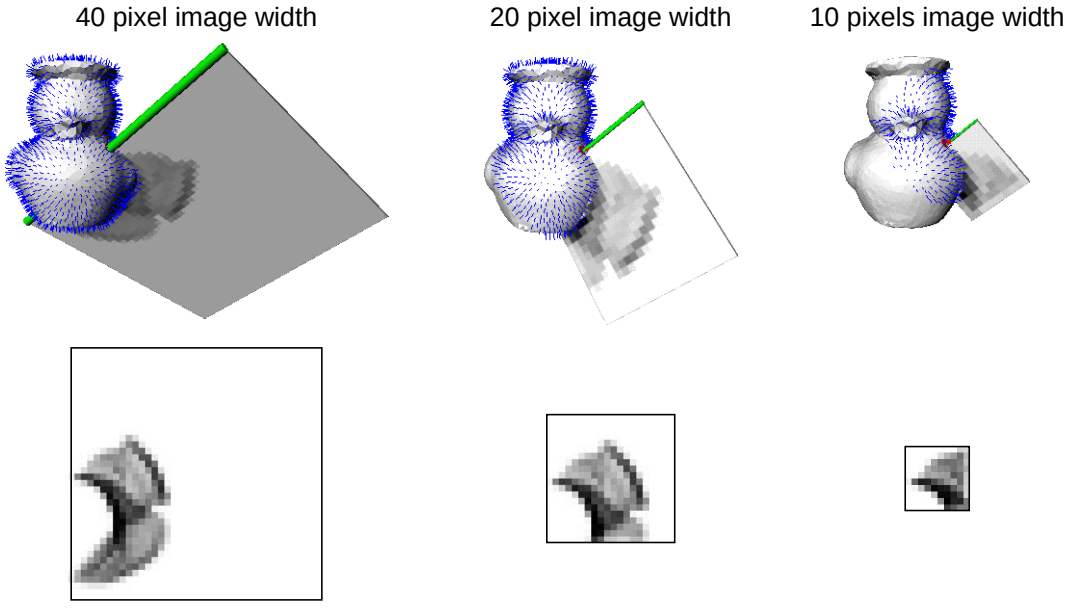


Figure 2-9: The effect of image width on spin-images. As image width decreases, the volume swept out by the spin-image(top) decreases, resulting in decreased spin-image support (bottom). By varying the image width, spin-images can vary smoothly from global to local representations.

has *W/2* rows above $\beta=0$ and *W/2* rows below $\beta=0$. Given the bin size, the bilinear weights used to increment the bins in the spin-image can be calculated

$$a = \alpha - ib \qquad b = \beta - jb. \tag{2.3}$$

The final spin-image generation parameter is *support angle* A_s. Support angle is the maximum angle between the direction of the oriented point basis of a spin-image and the surface normal of points that are allowed to contribute to the spin-image. Suppose we have an oriented point A with position and normal $(\boldsymbol{p}_A,\boldsymbol{n}_A)$ for which we are creating a spin-image. Furthermore, suppose there exists another oriented point B with position and normal $(\boldsymbol{p}_B,\boldsymbol{n}_B)$. The support angle constraint can then be stated as: B will be accumulated in the spin-image of A if

$$\operatorname{acos}(\boldsymbol{n}_A \cdot \boldsymbol{n}_B) < A_s. \tag{2.4}$$

Support angle is used to limit the effect of self occlusion and clutter during spin-image matching; this is discussed in more detail in Chapter 5. As discussed in Chapter 9, decreasing support angle also has the effect of decreasing the descriptiveness of spin-images. Generally, support angle is set between 90˚ and 60˚. Figure 2-10 shows the spin-image generated for three different support angles along with the vertices on the model that are mapped into the spin-image.

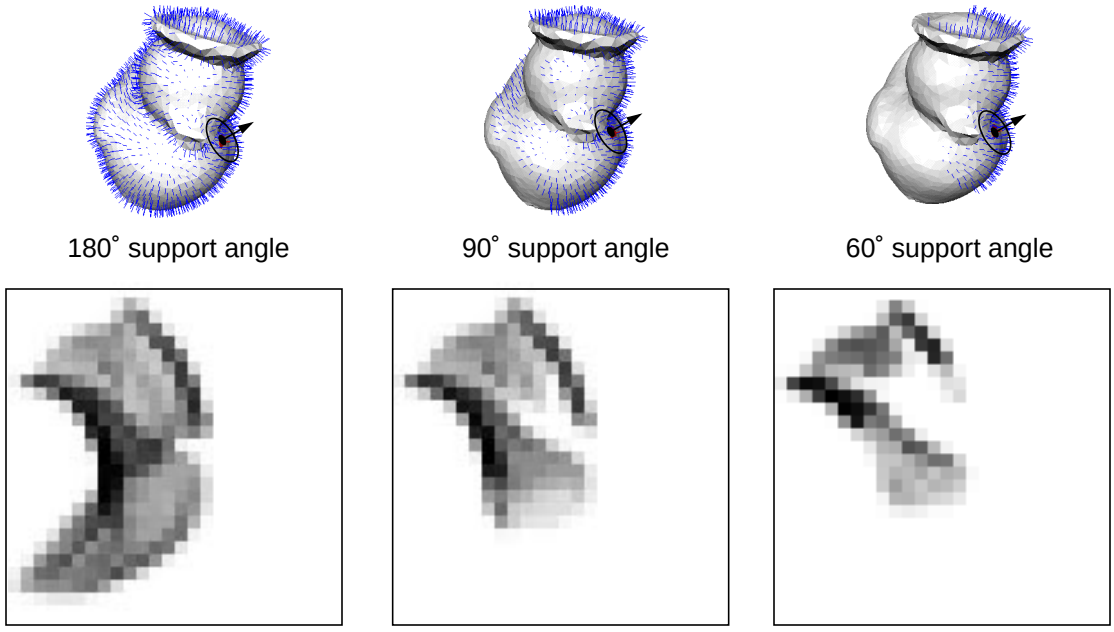


Figure 2-10: The effect of support angle on spin-image appearance. As support angle decreases, the number of points contributing to the spin-image (top) decreases. This results in reduction in the support of the spin-images (bottom).

2.4 Comparing Spin-Images

A spin-image is an object-centered (i.e., pose-independent) encoding of the shape of an object, because the spin-map coordinates of a point with respect to a particular oriented point basis are independent of rigid transformations. Suppose we have two instances of an object in two different poses. Because spin-images are pose independent, the two objects will have similar spin-images, if they are generated with the same spin-image generation parameters. By directly comparing spin-images from one object with spin-images from the other object, point correspondences can be established between points on one object and points on the other object that have the same spin-image. The point correspondences can then be used to localize one object with respect to the other. Implicit in this method of recognition is a method for comparison of spin-images.

Since spin-images from different instances of an object will not be exactly the same, a meaningful way of comparing two spin-images must be developed. We would like the method of comparison to be efficient and tailored to the image generation process. We expect two spin-images from proximal points on the surface of two different instances of an object to be linearly related because the number of points that fall in corresponding bins will be similar (given that the distribution of points over the surface of the objects is the same). Since the bin values are directly related to the number of points falling into the bins, the bin values will be similar. From template matching, the standard way of comparing linearly related images is the normalized linear correlation coefficient. Given two spin-images P and Q with N bins each, the linear correlation coefficient $R(P,Q)$ is

$$R(P, Q) = \frac{N\sum p_i q_i - \sum p_i \sum q_i}{\sqrt{(N\sum p_i^2 - (\sum p_i)^2)(N\sum q_i^2 - (\sum q_i)^2)}} \quad . \tag{2.5}$$

R is between -1 (anti-correlated) and 1 (completely correlated), and it measures the normalized error using the distance between the data and the best least squares fit line to the data. R provides a method for the comparison of two spin-images: when R is high, the images are similar; when R is low the images, are not similar. The correlation coefficient imposes an ordering on point correspondences, so good and bad correspondences can be differentiated.

The linear correlation coefficient provides a simple and established way to compare two spin-images that can be expected to be similar across the entire image. Such is the case when a spin-image from a complete object is being compared to a spin-image from a different, but complete, object. However, in many cases the 3-D data available is not complete (as in the case of range images taken from different views), and the spin-images from different incomplete data sets will not be the same everywhere in the images. For example, spin images generated from range images will have clutter (extra data) and occlusions (missing data).

A possible way to handle this case is to compare the images using a robust estimator such as least median of squares. However, we have found that the overall robustness of our method decreases the need for such complex and time consuming estimation. We have found a simpler occlusion handling method that involves removing portions of the images from the image comparison. If a bin in either of the images does not have a value (i.e., no vertex was spin-mapped into it), then that bin is not considered in the calculation of the linear correlation coefficient. In other words, the data used to compute the correlation coefficient is taken only from the region of overlap between two spin images. As shown in Chapter 5, clutter in the scene can create scene spin-image bins that have data where the model spin-image has no data. Similarly, occlusion in the scene can create scene pixels that have no data where the model spin-image has data. In this case, knowledge of the effect of clutter and occlusion on the spin-image generation process is used to eliminate outlier pixels from the correlation coefficient computation.

Since the linear correlation coefficient is a function of the number of pixels used to compute it, the amount of overlap between spin-images will have an effect on the correlation coefficients obtained. The more pixels used to compute a correlation coefficient, the more confidence there is in its value. If the correlation coefficient is used alone, then it is possible to have two images that have small overlap, but are similar only in the area of overlap, to have a higher correlation coefficient than two images that have large overlap, but are slightly less similar in the area of overlap. The two images with more overlap should be given a higher rank, because there is more confidence in the measured correlation coefficient.

We have found that matching of spin-images based on the magnitude of correlation coefficient as well as the confidence in the correlation coefficient results in a greater number of correct oriented point correspondences. The confidence in the correlation coefficient can be measured

by its variance. We use the similarity measure *C*, derived in detail in Appendix B, to combine correlation coefficient R and its variance into a single function. When comparing two spin-images *P* and *Q* where *N* is the number of overlapping pixels used in the computation of *R*, the similarity measure *C* is defined as

$$C(P, Q) = (\operatorname{atanh}(R(P, Q)))^2 - \lambda\left(\frac{1}{N-3}\right) \tag{2.6}$$

This similarity measure is a heuristic loss function that will return a high value for two images that are highly correlated and have a large number of overlapping bins. The change of variables, a standard statistical technique ([19] Chapter 12) performed by the hyperbolic arctangent function, transforms the correlation coefficient into a distribution that has better statistical properties. In particular, under the transformation of variables, the variance of the transformed correlation coefficient becomes *1/(N-3),* which is a simple function of the number of pixels used to compute *R*.

In (2.6), λ weights the variance against the expected value of the correlation coefficient. Analysis of (2.6) shows that λ controls the point at which overlap between spin-images becomes important for comparison of spin-images. When overlap is much larger that λ, the second term in (2.6) becomes negligible. In contrast when the overlap is much less than λ, then the second term dominates the similarity measure. Therefore, λ should be set close to the expected overlap between spin-images. In the next chapter, we give an algorithm for automatically setting λ based on the overlap between spin-images generated from a surface.

Figure 2-11 illustrates how spin-images are compared between two different (different points, different connectivity) surface meshes representing rubber duckie. One the left, a single oriented point *A* with associated spin-image is selected from duckie 1; on the left, two oriented points *B* and *C* with associated spin-images are selected from duckie 2. Points *A* and *B* are in similar positions on the object, so their spin-images are similar, as is shown by the scatter plot of the images. A scatter plot of the image data, created by plotting the pixel values in one image versus the corresponding pixels values in the other image, is a useful way of visualizing whether two data sets are correlated; the distribution of data points will cluster around a line when the two images are linearly correlated. For points *A* and *B*, the scatter plot of the image data shows

high correlation (R = 0.927) and overlap (N=289). Points *A* and *C* are not in similar positions on the duckie, so their spin-images are not similar. The scatter plot of the image data shows low correlation and (R = 0.066) and low overlap (N=171). This example validates that both high correlation coefficient and high data overlap indicate correct matches.

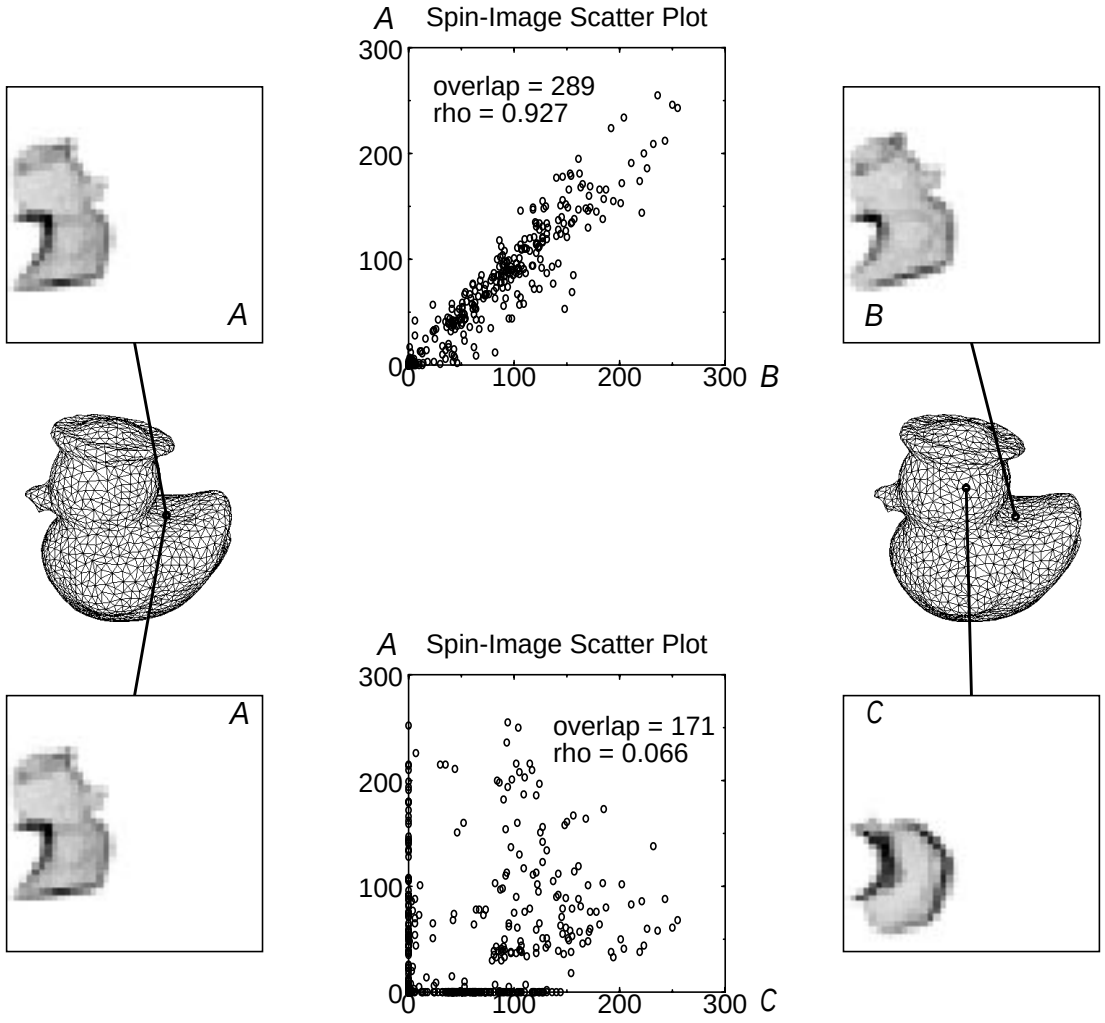


Figure 2-11: Comparison of spin-images visualized as scatter plots. When spin-images are similar, they are linearly correlated and have a high overlap. When spin-images are dissimilar, because they come from different positions on the surface mesh, they are not linearly correlated and have a low overlap.

2.5 Summary

In summary, spin-images are useful representations for describing surface shape for the following reasons.

- pose invariant / object-centered
- simple to compute / transformation of the data
- scalable from local to global representation
- robust to clutter and occlusion
- impose minimal restrictions on surface shape
- widely applicable to problems in 3-D computer vision

In the next chapter, we describe the surface matching engine which matches surfaces by matching oriented points using spin-images.

Chapter 3

The Surface Matching Engine

Surface matching is achieved by matching oriented points using spin-images. Spin-images from points on one surface are compared to spin-images from points on another surface; when two spin-images are similar, a point correspondence between the surfaces is established. Several point correspondences are then grouped using geometric consistency to calculate a transformation that aligns the surfaces. To verify surface matches, we have developed an efficient iterative closest point (ICP) verification algorithm which verifies matches and refines transformations. A schematic of surface matching is shown in Figure 3-1 and a detailed block diagram of the surface matching process is shown in Figure 3-2.

Our surface matching algorithm has many advantages. Since spin-images are generated at every point in the surface mesh, error prone feature extraction and segmentation are avoided.

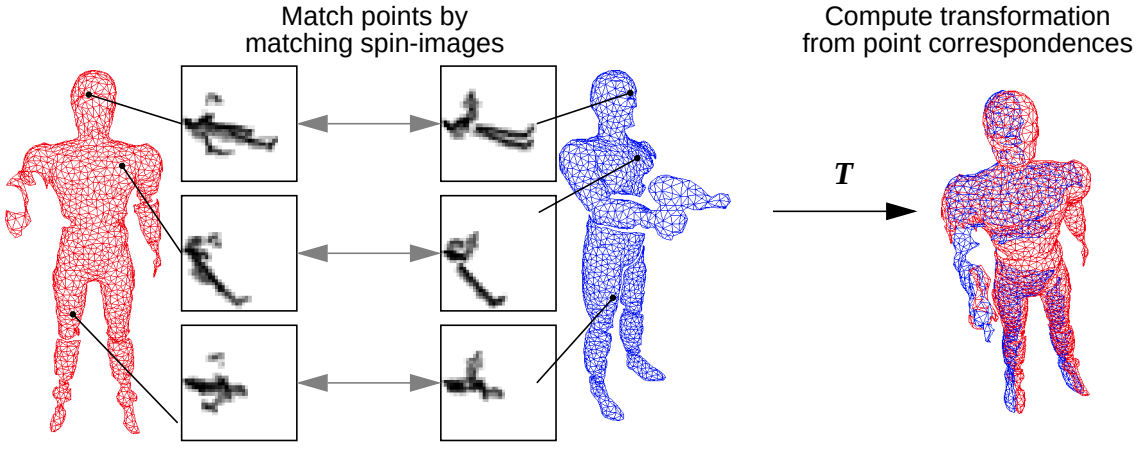


Figure 3-1: Surface matching using spin-images between two range views of a toy robot. First, point correspondences are established by matching spin-images. Next, point correspondences are grouped using geometric consistency, and a transformation that aligns the surfaces is computed. This transformation is subsequently verified and refined.

Since we make no assumptions about the topology or shape of the objects represented, our algorithm can match arbitrarily shaped surfaces, and, since no initial transformation is required, our algorithm can be used for surface matching when information about sensor position is unknown. Since many points are matched between surfaces, incorrect matches due to clutter and occlusion can be detected and eliminated.

To facilitate discussion of surface matching, we will use the following standard terminology. When matching two surfaces, one of the surfaces will be called the scene and the other surface will be called the model. When matching surfaces for surface registration, there exists little distinction between model and scene. However, when using surface matching to recognize objects, the scene corresponds to the sensed data and the model corresponds to a predetermined object to be matched to the scene. Using this convention, transformations computed from sur-

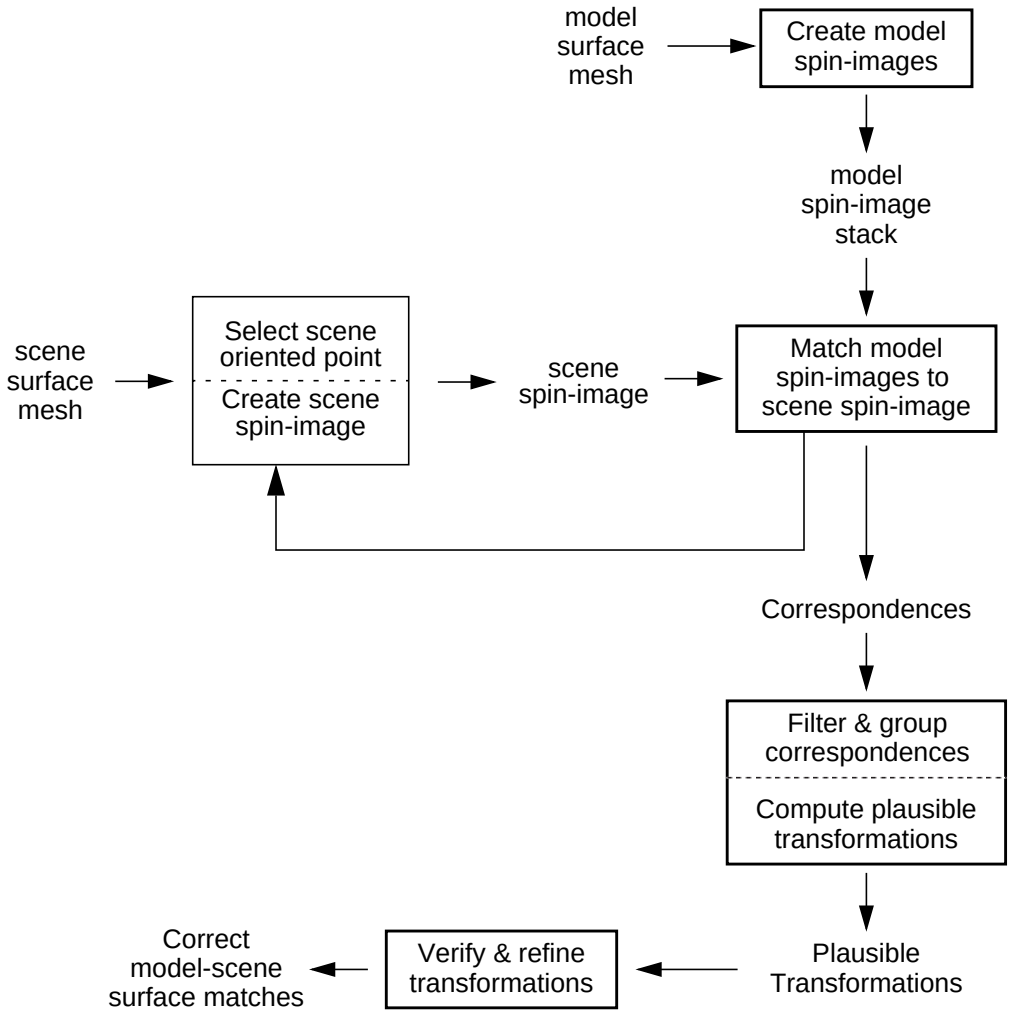


Figure 3-2: Surface matching block diagram.

face matching map the model to the scene.

3.1 Spin-Image Matching

The similarity measure (2.6) gives us a way of ranking matches between spin-images; two spin-images with a high similarity measure are likely to come from corresponding points. With a way of comparing and ranking spin-image pairs based on similarity measure, a method for establishment of point correspondences between oriented points in a model and a scene data set can be been developed. First, spin-images are generated for all points on a model surface mesh, and then these images are stored in a *spin-image stack.* Next, a scene point is selected randomly from the scene surface mesh and its spin-image is generated. The scene spin-image is then correlated with all of the images in the model spin-image stack and the similarity measures (2.6) for each image pair are calculated. These similarity measures are stored in a similarity measure histogram. Upper outliers in the similarity measure histogram correspond to model/scene pairs with similarity measures that are significantly higher than the rest of the model/scene pairs. From these outliers, plausible model/scene *point correspondences* are between their associated oriented points. This procedure to establish point correspondences is then repeated for a fixed number of randomly selected scene points. The end result is a set of model/scene point correspondences which can be filtered and grouped in order to compute transformations from model to scene.

Comparison of a scene spin-image with all of the model spin-images can be viewed as an effi-

```
SpinImageMatching(surface_mesh MODEL, surface_mesh SCENE)
    For all oriented points m on MODEL                                  // make spin-image stack
        CreateSpinImage(m,MODEL_SPIN_IMAGE,MODEL)
        add MODEL_SPIN_IMAGE to MODEL_STACK
    SelectRandomScenePoints(SCENE_POINTS,SCENE)                         // select scene points
    for all oriented points s in SCENE_POINTS                           // compare with model
        CreateSpinImage(s,SCENE_SPIN_IMAGE,SCENE)
        ComputeSimilarityMeasures(MEASURES,SCENE_SPIN_IMAGE,MODEL_STACK)
        CreateSimilarityMeasureHistogram(HISTOGRAM,MEASURES)
        DetectOutliers(OUTLIERS,HISTOGRAM)
        for all model points m in OUTLIERS
            CreatePointCorrespondence(s,m)
            add correspondence [s,m] to list of possible point correspondences
```

Figure 3-3: Pseudo-code description of the spin-image matching algorithm for establishing point correspondences between a model and scene surface mesh.

cient convolution of a portion of the scene with the model. Every point in the model is compared to the scene points using the correlation of the model and scene spin-images. This comparison does not require the calculation of a transformation. A true 3-D convolution of the scene and the model would require the discretization of six dimensional pose space and then the pointwise comparison of scene to model for each transformation in the discretized pose space. 3-D convolution is too computationally intensive to be feasible.

In the last chapter, we mentioned that the parameter λ, used when calculating the similarity measure between two spin-images, could be automatically set based on properties of a set of spin-images. Now we have enough information to explain how this is done. We want λ to limit matches between spin-images of low overlap. To do this we need to know what the expected overlap between correctly matched spin-images between two surfaces will be. We can use the spin-images from the model surface mesh to compute λ before any comparison of spin-images occurs. For each model spin-image, count the number of bins that have data. Put these values, for all the spin-images in the model, in a list and determine the median value of the list. This median value is a reasonable measure of the expected overlap between spin-images. To allow matches with smaller overlap, due to occlusion, we set λ to one half of this value.

A pseudo-code description of the algorithm for establishing a set of point correspondences between model surface mesh and a scene surface mesh is given in Figure 3-3. The primary issues that still need to be addressed when establishing point correspondences are: how to select scene points, and how to determine correspondences with large similarity.

3.1.1 Selecting Scene Points

In our current implementation, we select a small number of scene points using random sampling of the indices of the scene points, in order to reduce the running time of our algorithm. Typical ordering of points from 3D sensors (by raster for range images, by contour for CT, by voxel for marching cubes) will give us a random sampling over the surface of the scene. In future implementations, we plan to select scene points in a more intelligent fashion by comparing spin-images of scene points and using only the spin-images that are significantly different from those already selected. This method will use the shape encoded in the spin-images to ensure a homogeneous sampling of points over the surface of the scene. The number of scene points se-

lected depends on the number of points and the amount of clutter in the scene. If there is no clutter, then all of the scene data corresponds to the model, so only a few (~10%) scene points need to be selected. On the other hand, if the scene is cluttered, then the number of scene points should adequately sample the entire surface of the object so that at least three scene points can be expected to lie on the portion of the scene data that corresponds to the model. Since this number cannot be directly measured, we set the number of scene points investigated to a fraction of the total number of scene points. In practice, we set this fraction to one half to one twentieth of the total number of scene points.

Another criterion that could be used for selecting scene spin-images is based on the accuracy of the scene surface normals. When the scene surface normals are inaccurate, spin-images that match the model cannot be generated. By selecting scene spin-images with accurate surface normals, the chances of scene spin-images matching model spin-images becomes more likely. We compute surface normals by fitting a plane to mesh vertices, so the residual error of the plane fit can be used as an estimate of the accuracy of the surface normal. We have implemented a method for scene point selection based on planar fit error, but have not conducted experiments to see if it results in significant improvement in matching performance.

3.1.2 Automatic Selection of Matching Model Points

A simple way to select the best model points matching the current scene point would be to select the top percentage of model points. However, this method is too naive and may select model points that are not appropriate matches given the similarity information. A better way of selecting the corresponding model points is by analyzing the histogram of similarity measures and selecting the model points that have similarity measures that are much higher than the rest. Essentially, good matches between the scene point and the model points can be determined by finding the upper outliers in this histogram because upper outliers correspond to points that match the scene points very well. If no outliers exist, then the scene point has a spin-image that is very similar to all of the model spin-images, so definite correspondences with this scene point cannot be established. If corresponding model points were selected as the top percent of model points in the similarity histogram, this situation would not be detected and unnecessary or incorrect correspondences would be established.

Histograms of similarity measure will typically have a single mode corresponding to incorrect spin-image matches. Correct spin-image matches will be upper outliers in the histogram of similarity measure. For well behaved distributions (i.e., single mode), a standard statistical way of detecting outliers is to determine the fourth spread of the histogram (f_s = upper fourth - lower fourth = median of largest N/2 measurements -median of smallest N/2 measurements)[19]. Statistically moderate outliers are $1.5f_s$ units above (below) the upper (lower) fourth, and extreme outliers are $3f_s$ units above (below) the upper (lower) fourth. Figure 3-4 shows a histogram of similarity measures. Four extreme outliers (points with similarity measure greater than $3f_s$ above the upper fourth) are detected. Through detection of extreme outliers in the histogram of similarity measures, an automatic method for establishing correspondences has been created.

A schematic description of matching a single scene spin-image to the spin-images of a model surface is given in Figure 3-5. Matching multiple scene spin-images to the spin-images of the model surface results in a list of plausible point correspondences between model and scene. The next section describes how the number of correspondences in the list is reduced by filtering of correspondences.

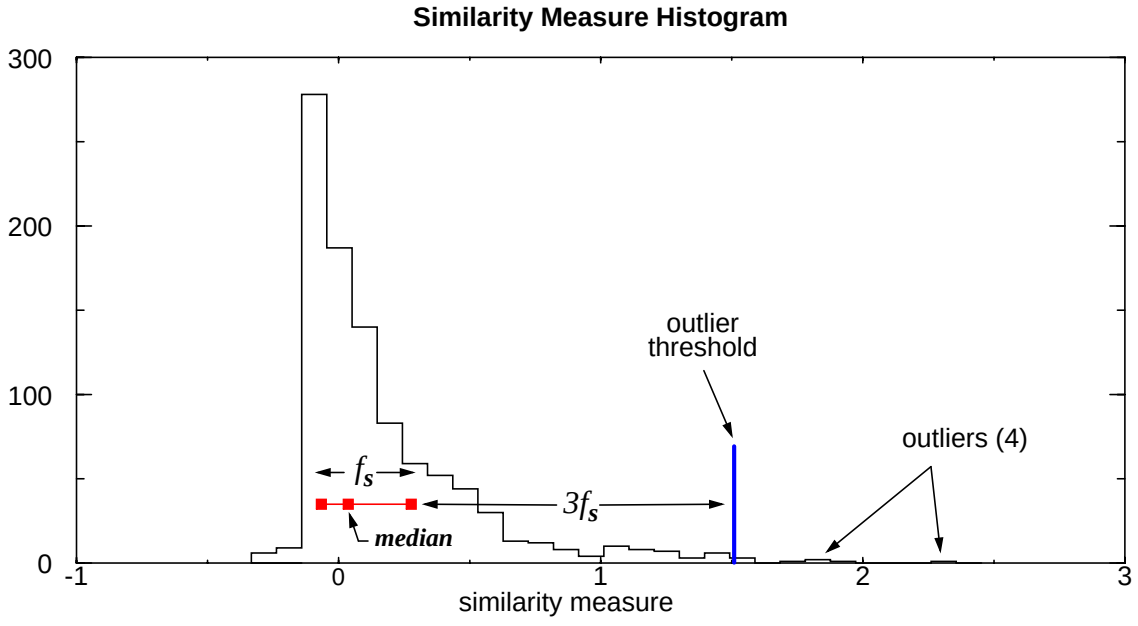


Figure 3-4: Similarity measure histogram comparing a scene point to all model points. Outliers in the similarity measure histogram correspond to model points that might match the scene point.

3.2 Correspondence Filtering

In order to establish a transformation from model to scene, at least three point correspondences need to be established. Spin-image matching drastically reduces the number of possible point correspondences by associating only scene and model points that have similar spin-images. However, the function still finds multiple correspondences between scene and model oriented points.

During matching, a single point can be matched to more than one point for two reasons. First, symmetry in the data and in spin image generation may cause two points to have similar spin-images. Second, spatially close points may have similar spin-images. Furthermore, if an object appears multiple times in the scene, then a single model point will match multiple scene points.

During matching, some points selected from parts of the scene that do not overlap the model surface may be incorrectly matched to model points. In general, these incorrect correspondences will have low similarity measures and be geometrically inconsistent when compared to the

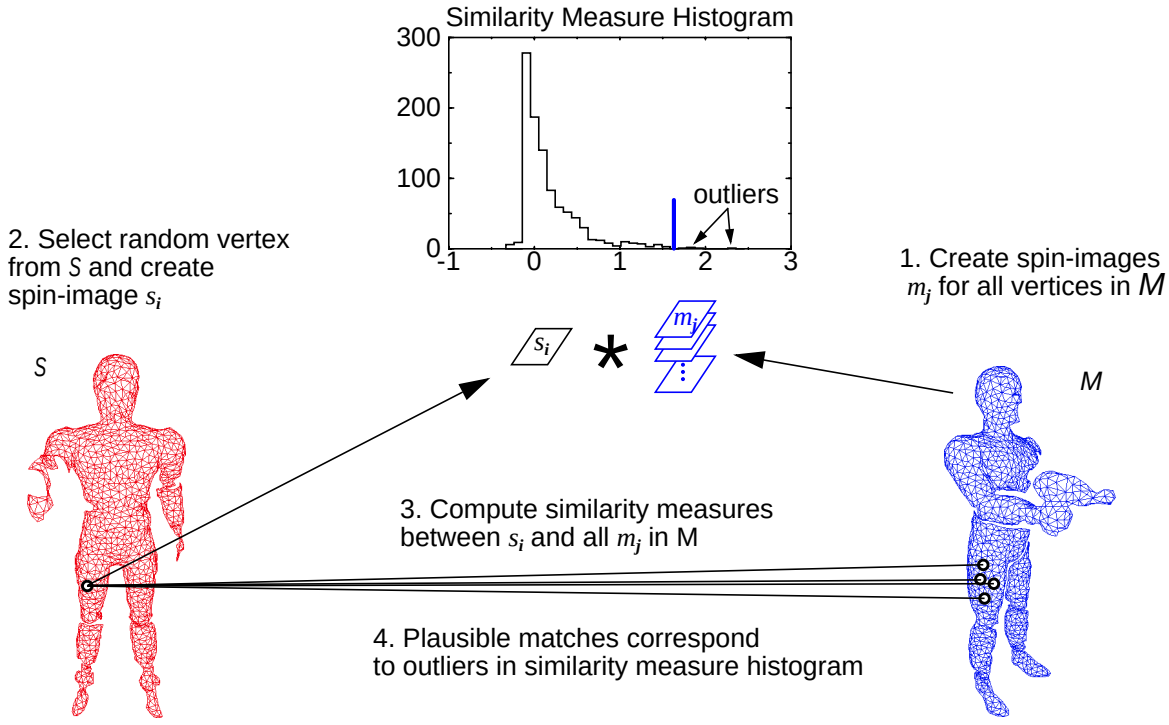


Figure 3-5: Matching a single spin-image from scene S to model M. First, all spin-images for all model oriented points are created, then a scene point is randomly selected and its spin-image is created. Next the similarity measures between the scene spin-image and all of the model spin-images are computed and put in a histogram. Finally, hypothesized model points corresponding to the scene point are detected by finding outliers in the similarity measure histogram.

rest of the correspondences. Given the numerous correspondences, it is possible to filter out bad correspondences based on properties of the correspondences taken as a group. This integral approach is robust because it does not require reasoning about specific point matches to decide which correspondences are the best. This approach is in contrast to hypothesize and test and alignment paradigms of recognition [34][95] where the minimal number of correspondences required to match model to scene are proposed and then verified through some other means.

The first method of correspondence filtering uses similarity measure to remove unlikely correspondences. Consider the list of plausible correspondences from spin-image matching. Find the correspondence in this list with maximum similarity measure. Next, remove from the list all correspondences with similarity measure that are less than some fraction of the maximum similarity measure. In practice, this fraction is set to one half. By using the maximum similarity measure to set the threshold on kept correspondences, a property of the group of correspondences is used to filter out unlikely correspondences.

The second method for filtering out unlikely correspondences uses geometric consistency which is a measure of the likelihood that two correspondences can be grouped together to calculate a transformation of model to scene. If a correspondence is not geometrically consistent with other correspondences, then it cannot be grouped with other correspondences to calculate a transformation, and it should be eliminated. The geometric consistency of two correspondences $C_1 = [\mathbf{s_1},\mathbf{m_1}]$ and $C_2 = [\mathbf{s_2},\mathbf{m_2}]$ is measured by comparing the spin-map coordinates (2.1) of corresponding points (Figure 3-6).

$$d_{gc}(C_1, C_2) = \frac{\left\| S_{m_2}(\mathbf{m}_1) - S_{s_2}(\mathbf{s_1}) \right\|}{(\left\| S_{m_2}(\mathbf{m_1}) \right\| + \left\| S_{s_2}(\mathbf{s_1}) \right\|)/2}$$
$$D_{gc} = max(d_{gc}(C_1, C_2), d_{gc}(C_2, C_1)) \tag{3.1}$$

$d_{\mathbf{gc}}$ measures the distance between the correspondences in spin-map coordinates normalized by the average length of the spin-map coordinates. Spin-map coordinates are used to measure geometric consistency because they are a compact way to measure consistency in position and normals. Distance between spin-map coordinates is normalized by average spin-map coordinates of the correspondences so that the geometric consistency measure is not biased toward correspondences that are close to each other. Since d_{gc} is not symmetric, the maximum of the dis-

tances is used to define the geometric consistency distance D_{gc}. When D_{gc} is small (i.e., C_1 and C_2 are geometrically consistent), the scene and model points in C_1 are the same distance apart and have the same angle between surface normals as the scene and model points in C_2.

To filter correspondences based on geometric consistency, the following procedure is applied to the list L of correspondenced remaining after filtering based on similarity measure. First, a geometric consistency threshold T_{gc} is selected such that if $D_{gc}(C_1,C_2) < T_{gc}$ then C_1 and C_2 are geometrically consistent. Usually T_{gc} is set to 0.25 to enforce strong geometric consistency between correspondences. Next for each C_1 in L, $D_{gc}(C_1,C_2)$ is computed with all of the other C_2 in L. If the number of correspondences C_2 where $D_{gc}(C_1,C_2) < T_{gc}$ is at least one quarter of the number of correspondences in L, then C_1 passes the geometric consistency test. Otherwise, C_1 is not geometrically consistent with enough correspondences in L, so it is removed from the list of point correspondences. This process is repeated for all C_1 in L.

The end result after filtering on similarity measure and geometric consistency is a list of point correspondences $L = \{C_1,..., C_n\}$ that are the most likely to be correct. In practice this number is between 20 and 50 correspondences. Figure 3-1 shows three of the best correspondences and matching spin-images between two views of a toy robot. The next step is to group these correspondences into sets that can be used to compute transformations.

3.3 Grouping Point Matches with Geometric Consistency

Single correspondences cannot be used to compute a transformation from model to scene be-

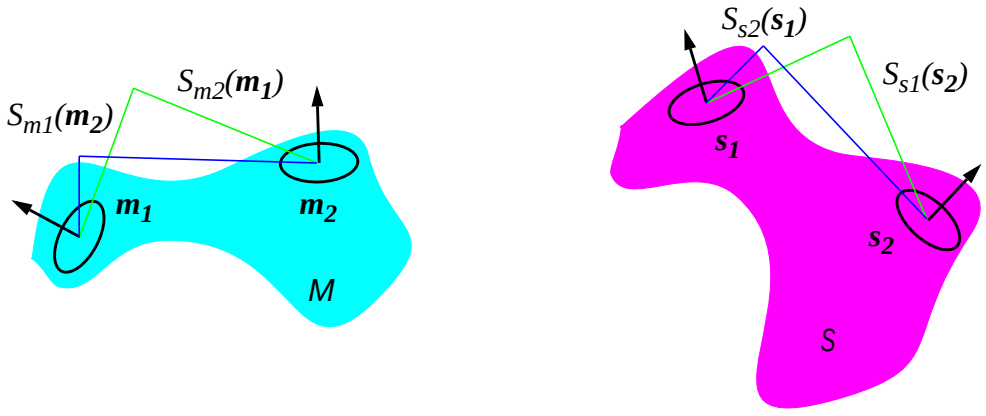


Figure 3-6: Spin-map coordinates between model oriented points and scene oriented points. When normalized spin-map coordinates between two oriented point correspondences are similar, the correspondences are geometrically consistent.

cause an oriented point basis encodes only five of the six necessary degrees of freedom. At least two oriented point correspondences are needed to calculate a transformation if position and normals are used. To avoid combinatoric explosion, geometric consistency is used to determine a groups of correspondences from which plausible transformations can be computed. Since many correspondences are grouped together and used to compute a transformation, the resulting transformation is more robust than one computed from a few correspondences.

We group correspondences based on a grouping criterion W_{gc} that is the geometric consistency distance between two correspondences (3.1) augmented by a weight that promotes grouping of correspondences that are far apart.

$$w_{gc}(C_1, C_2) = \frac{d_{gc}(C_1, C_2)}{1 - e^{-\left(\left\|S_{m_2}(\boldsymbol{m_1})\right\| + \left\|S_{s_2}(\boldsymbol{s_1})\right\| / (2\gamma)\right)}}$$
$$W_{gc}(C_1, C_2) = max(w_{gc}(C_1, C_2), w_{gc}(C_2, C_1)) \tag{3.2}$$

W_{gc} will be small when two correspondences are geometrically consistent and far apart. Since correspondences that are not geometrically consistent will produce transformations of high error, geometric consistency is a necessary condition for grouping. Correspondences whose points are close together generate transformations that are susceptible to noise in point position [14], so correspondences with points that are spread apart are more desirable. γ is used to normalize the geometric consistency weight to make it scale independent. γ is fixed at four times the mesh resolution; since γ never changes, it is not considered a parameter in our matching algorithm. Setting γ in this way encourages grouping between correspondences that are at least four times the mesh resolution distance from each other.

The grouping criterion between a correspondence C and a group of correspondences $\{C_i,..., C_n\}$ is

$$W_{gc}(C, \{C_1, \ldots, C_n\}) = \max_i (W_{gc}(C, C_i)) \tag{3.3}$$

In other words, the grouping criterion between a correspondence and a group of correspondences will be small if the correspondence is geometrically consistent with, and far apart from, all of the correspondences in the group.

Given a list of most likely correspondences $L = \{C_1,..., C_n\}$ the grouping procedure that is ap-

plied for each correspondence in the list is as follows: Select a seed correspondence C_i in L and initialize a group $G_i = \{C_i\}$. Find the correspondence C_j in L, for which $W_{gc}(C_j,G_i)$ is a minimum. Add C_j to G_i if $W_{gc}(C_j,G_i) < T_{gc}$ where the threshold T_{gc} is set between zero and one. Continue adding the correspondence with minimum grouping criterion until no more correspondences can be added to G_i. The grouping procedure is performed for each correspondence in L, and the end result is n groups, one for each correspondence in L. T_{gc} is usually set to 0.25; this threshold will group correspondences that are very geometrically consistent.

This grouping algorithm allows a correspondence to appear in multiple groups, a condition which is necessary to handle model symmetry. For example, the CAD model in Figure 2-3 has a plane of symmetry resulting in two feasible transformations. Correspondences along the plane of symmetry contribute to two distinct transformations.

A plausible rigid transformation T from model to scene is calculated from each group $\{[\boldsymbol{m}_i,\boldsymbol{s}_i]\}$ of correspondences by minimizing

$$E_T = \sum \left\| \boldsymbol{s}_i - T(\boldsymbol{m}_i) \right\|^2 . \tag{3.4}$$

Instead of using points and normals in the minimization of (3.4), we use only the position of the oriented points. This allows us to use a well defined algorithm for finding the best rigid transformation that aligns two point sets [26][45]. The transformations and associated correspondence groups are then input into a verification procedure.

Our method for grouping correspondences can be related to interpretation tree methods of object recognition [33][34][11] where geometric consistency is used to find feature correspondences between two data sets. However, in our approach, the unary constraints imposed by spin-image matching greatly reduce the possible matches between features. Therefore, for grouping correspondences, we can use a much less sophisticated technique than the interpretation tree. Furthermore, our method does not stop, as in alignment forms of recognition [46], when a sufficient number of correspondences to calculate a transformation are found. Instead, we search for many point correspondences between scene and model thereby increasing the likelihood of transformations computed and allowing for multiple matches between scene and model. This is necessary when using spin-image matching for object recognition where multiple instances of an object can occur in the scene.

It is still possible that incorrect correspondences pass through correspondence grouping resulting in incorrect transformations of model to scene. However, the final verification stage of the surface matching engine (discussed in the next section) will effectively eliminate these transformations. If outlier correspondences were truly an issue in matching (in our experience they are not) then robust clustering techniques (e.g., RANSAC) could be used to compute correspondence groups.

3.4 Verifying Matches

The purpose of verification is to find the best match(es) between model and scene by eliminating matches that are inconsistent when all of the scene data is compared to all of the model data. Ideally the verification step should be efficient, never eliminate good matches, and refine the transformation of model to scene if possible. Our verification algorithm is a formulation of the Iterative Closest Point (ICP) algorithm [7][12][81][100] that can handle partially overlapping point sets and arbitrary transformations. During verification, point correspondences are spread over the surfaces of the scene and the model from the initial correspondences established by matching of spin-images. If many correspondences are established through spreading, a match between model and scene is validated. Not only does this method verify possible matches, but it also improves the transformation between scene and model.

The ICP algorithm concurrently proposed by Besl & McKay [3] and Zhang [100] is an excellent method for the registration of free form curves and surfaces when the transformation between model and scene is small. The algorithm iteratively determines the transformation between a model and a scene by assigning correspondences between closest points, calculating the transformation, transforming all of the model points based on the new transformation and then repeating the process. The ICP algorithm works well even in the presence of large amounts of noise when the initial transformation is small. Unfortunately, because the algorithm converges to the nearest local minimum in pose space, it cannot be used when the model and scene are arbitrarily displaced from each other. Another problem with the generic form of ICP is that it has difficulty registering data sets when one is not a subset of the other because it tries to establish correspondences between all the points in one set with some of the points in the other. Our verification algorithm is a formulation of ICP that can handle partially overlapping point

sets and arbitrary transformations because it starts with a transformation generated from matching spin-images.

Given a coarse match between model and scene, the purpose of verification is to determine if the match is a good one and, if possible, to improve the transformation. The usual procedure is to measure some distance between the scene and the model; a good match will have a small distance. A simple way to measure the distance when the model and scene are composed of points is to find the closest scene point to every point in the model and take the average distance. Difficulties occur (as stated above) when the sets only partially overlap. Methods are needed to limit the closest point distance measurement only to those areas in the two sets that overlap. Our verification algorithm does this by growing closest point correspondences from initial correspondences established by spin-image matching.

Two surfaces match, or overlap, when many points on one surface are close to many points on the other surface. If the surfaces are oriented, then matched surfaces should also have consistently oriented surface normals; closest points between the two surfaces should be similar in 3-D position and surface normal. To account for consistency in surface normals, our verification algorithm measures closest point distances by using six-dimensional distance that combines position and surface normal. Given two oriented points $(\boldsymbol{p_1},\boldsymbol{n_1})$ and $(\boldsymbol{p_2},\boldsymbol{n_2})$ the 6-D distance between the points is

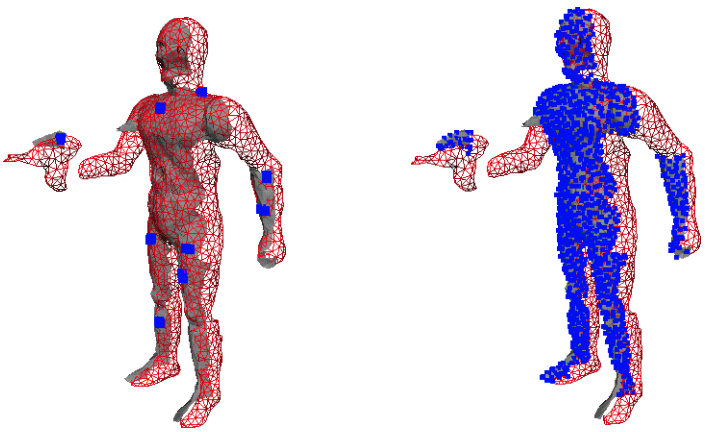


Figure 3-7: Transformation verification of two views of a toy robot.On the left is shown the match between the views before verification, one surface is shown as wireframe, the other is shown shaded; initial correspondences from spin-image matching are shown as squares. On the right is shown the same view of the match after verification with spread correspondences shown as smaller squares. Notice that correspondences are not established outside the region of overlap between the two views.

$$d_6 = \sqrt{\|\boldsymbol{p}_1 - \boldsymbol{p}_2\| + \nu\|\boldsymbol{n}_1 - \boldsymbol{n}_2\|} \tag{3.5}$$

where ν weighs surface normal information against position information.

Verification starts with an initial set of point correspondences from which the transformation of model to scene is computed. This transformation is then applied to the model surface mesh. Next, correspondences are spread from each initial correspondence as follows: For each scene point in an initial correspondence, the scene points directly connected to it by edges in the scene surface mesh are turned into correspondences (with their closest model points in 6D space) if the distance between scene and closest model points is less than a threshold D_v. This process is recursively applied to each of the correspondences just added until no more correspondences can be created. This spreading of correspondences is shown schematically in Figure 3-7.

This verification algorithm spreads correspondences between the model and the scene from the initial correspondences, and a cascade effect occurs. If the initial transformation is correct, a large number of points will be brought into correspondence; if the transformation is bad, the number of correspondences will remain close to the original number. Therefore, our measure of the validity of the match is the number of correspondences after verification which is related to the total aligned surface area between the model and scene by the transformation. If the number of correspondences after verification is greater than one tenth the total number of points on the model, the transformation is considered valid. Other matching metrics used in surface registration are discussed at length by Simon [82].

To speed up the search for closest model points, a six dimensional k-D tree [5][31] is generated for oriented points in the model surface mesh before the spreading of point correspondences. To do so, each model vertex is transformed into a 6-D point using the coordinates $(\boldsymbol{p}, \nu\boldsymbol{n}) = (p_x, p_y, p_z, \nu n_x, \nu n_y, \nu n_z)$. The k-D tree is constructed from these coordinates using the algorithm described by Zhang [100]. The parameter ν, controls the dominance of surface normal in the distance between 6-D points. We want surface normal to have an effect that is on order of the variation between 3-D positions in the mesh, so we set ν to a multiple of model mesh resolution. In our code, ν is hard coded to two times the mesh resolution, causing normals of vertices to have more of an effect on 6-D distance than the positions of the vertices have.

The threshold D_v in the verification stage (that sets the maximum 6-D distance by which two points can differ and still be brought into correspondence) is set automatically to two times the resolution of the meshes. This threshold allows for noise but prevents establishment of correspondences in regions where the data sets do not overlap because of different surface normals or positions.

If a match between the two surfaces has been established, then the number of correspondences between model and scene is larger than that obtained through spin-image matching. We use these additional correspondences to refine the transformation, after it has been verified and accepted, using a 6-D ICP registration algorithm that uses normals and positions [9][97]. The final transformation computed from the ICP algorithm is more accurate than the transformation computed through matching of spin-images because a large number of scene points are registered to model points. Considered as an ICP registration algorithm, our verification algorithm solves two of the biggest problems with ICP: First, the need for an initial guess at the transformation between model and scene is provided by the transformation determined by matching spin-images. Second, registering data sets when one is not a proper subset of the other is handled by the controlled spreading of correspondences from initial correspondences.

There are two results from surface matching, a transformation that aligns two surfaces and an estimate of the similarity of two surfaces based on their overlap. The transformation that aligns the surfaces is sufficient for determining the overlap of the surfaces; however, it can be made more accurate through application of more sophisticated surface registration techniques. For example, using the transformation supplied by surface matching as an initial estimate, an ICP algorithm could be applied to a finer resolution representation of the surface to obtain more accurate registration. In the next chapter, we show how surface matching, followed by refinement using ICP, results in accurate registration of range images for object modeling. In addition to using a finer resolution, more robust distance metrics like the Lorentzian [97] could also be employed to improve the alignment of the surfaces.

3.5 Results

This section describes some surface matching results from many different sensing domains and applications. Surface matching has two main applications: surface registration for determining

the transformation between two views of a scene or object and object recognition for deciding which objects in an object library exist in a scene. Object modeling, an important application of surface registration, is discussed in detail Chapter 4. Using surface matching for object recognition is discussed in Chapters 5 through 8. The following results hint at the broad range of applications of surface matching; more extensive results are given in the following chapters.

An important application domain for surface matching is interior modeling. In interior modeling, object surfaces are matched to range images of complex industrial interiors. A complete interior modeling system that uses spin-images to match surfaces is described in detail in Chapter 8. Figure 3-8 shows the recognition of a valve model in a cluttered industrial scene. (Figure 8-9 gives additional results in Chapter 8). The surface mesh model of the valve was generated from a CAD drawing using finite element software to tessellate the valve surface. The scene image was acquired with a Perceptron 5000 scanning laser rangefinder. The scene data was processed by removing long edges associated with step discontinuities, applying a "smoothing without shrinking" filter [88], and then applying our control of mesh resolution algorithm explained in Appendix A. In this example, the scene data is complex with a great deal of clutter. Furthermore, all the model exhibits symmetry which makes surface matching more difficult, because a single scene point can match multiple model points. This results clearly

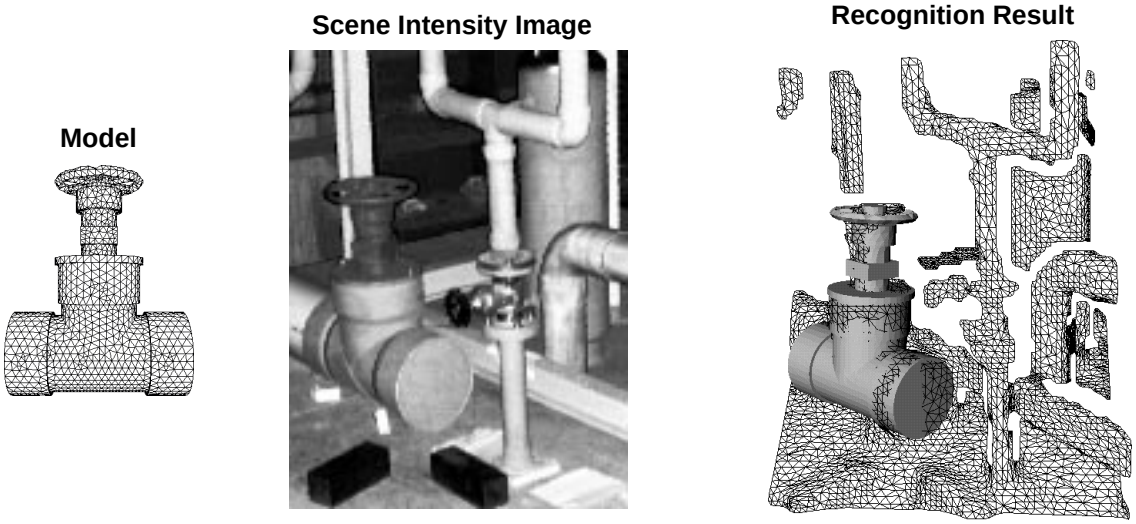


Figure 3-8: The recognition of an industrial object in a complex scene. On the left is shown a wireframe model of a valve created from a CAD drawings using finite element software for surface tessellation. In the middle is shown the intensity image acquired when a scanning laser rangefinder imaged the scene. On the right is shown the valve model (shaded) superimposed on the scene data (wireframe). These results demonstrate the recognition of complicated symmetric objects in 3-D scene data containing clutter.

show the ability of our algorithm to match complete 3-D model surfaces to partial 3-D scenes.

Another important task in interior modeling is the automatic registration of range images. By registering and merging range images, more complete scene descriptions are generated. Our matching algorithm provides a technique for determining the transformation between range views when it is unknown or highly uncertain. Figure 3-9 shows a scene composed of a PVC pipe joint, four graphite bricks, a piece of plywood and a steel valve placed on a mobile cart. This scene was imaged in three different positions by moving the cart and taking a range image with a Perceptron 5000 laser rangefinder at each position. The position of the cart varied each time by approximately 30 degrees of rotation about the vertical axis. Figure 3-9 shows the intensity channel of the range scanner for the three scene positions, a shaded view of the resulting registrations and an oblique and top view of the points in the three scenes after registration. The top view of the registered points clearly shows that a more complete scene description than is available from one range image is generated and that the registration is correct up to the resolution of the points. Additional range image registration results are given in Chapter 4 which discusses an object modeling system used to create complete models of object from partial range views.

Figure 3-10 demonstrates the matching of two models of free-form shape to the same surface. The femur and pelvis model were acquired using a CT scan of the bones, followed by surface

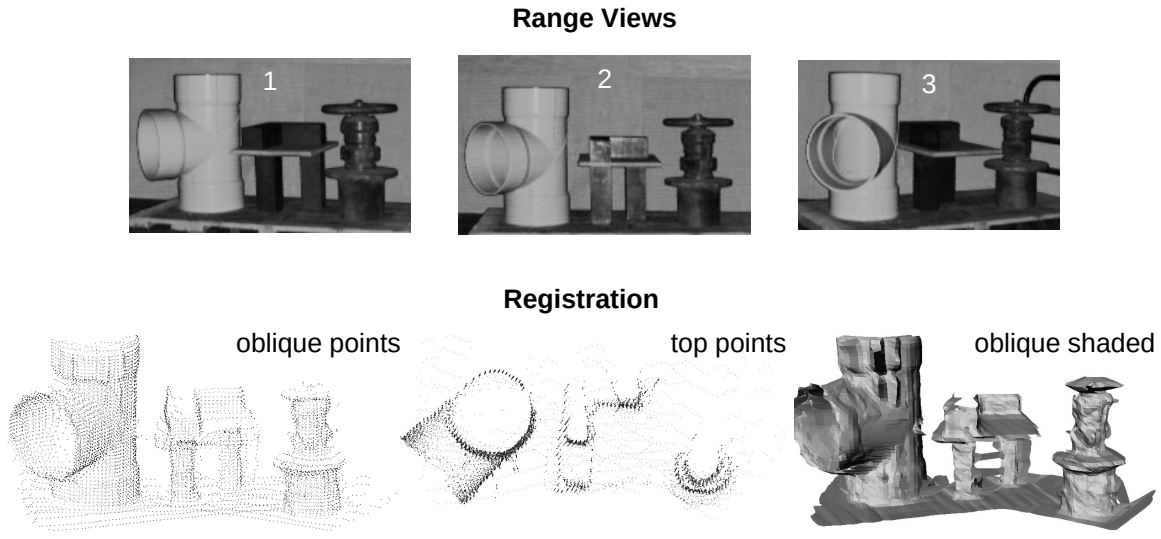


Figure 3-9: Registration of laser rangefinder images of industrial objects. The range views differ by roughly 30 degrees, so registering the views provides a more complete description of the scene.

mesh generation from contours. The scene was acquired using a K^2T structured light range camera. Before matching, the scene mesh is processed to remove long edges and small surface patches and then smoothed. Our algorithm was able to match the surfaces even in the presence of extreme occlusion; only 20% of the surface of the pelvis is visible. This result also demonstrates the matching of surfaces sensed with different sensing modalities.

A common result in the biomedical imaging literature is the registration of two skull data sets generated from volumetric medical scanners [36]. Figure 3-11 shows the registration of two different surface meshes of a skull created from the same CT data set. The surface meshes were created by first adding small amounts of Gaussian noise to the points in the original surface mesh generated from the CT scans and then simplifying the meshes. Different random noise values were added to the original surface mesh for each of the meshes shown, resulting in sur-

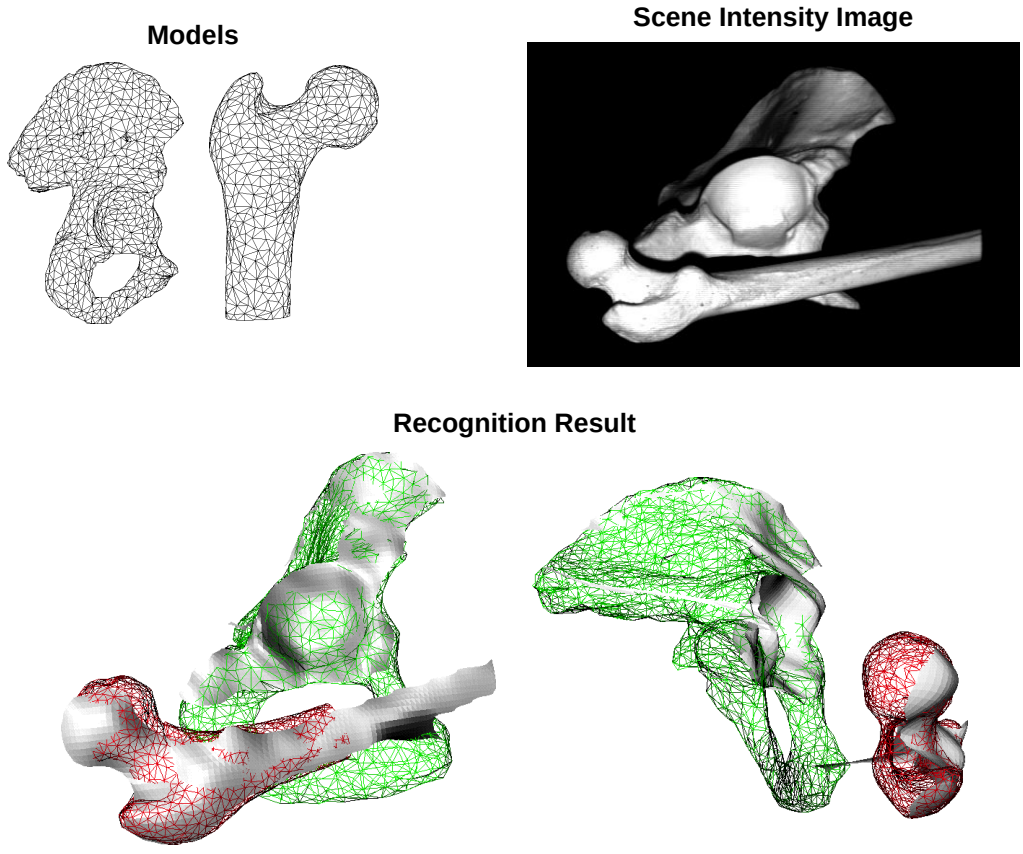


Figure 3-10: Matching of a femur and a pelvis bone in a range image. The femur and pelvis surfaces are matched even in the presence of extreme occlusion (only 20% of the pelvis surface is visible) and clutter. Shown are the pelvis and femur models acquired with CT, the scene image from a structured light range camera, and then two views of the recognition results where the scene data is shaded and the models are in wireframe. This result demonstrates simultaneous matching of free-form objects acquired using different sensing modalities.

face meshes with different points and connectivity after simplification. A close look at the two wireframe data sets in Figure 3-11 shows that the two surface meshes are completely different while still approximating the shape of the skull. This skull data set is especially difficult to register because the inner and outer surface of the cranium are extracted from the CT data, increasing the complexity of the model and possibly introducing ambiguities when registering the data sets. Since the two data sets are already co-registered, any non-identity transformation calculated for the registration will represent the registration error. For the result shown in Figure 3-11, the translation is $[0.019\ -0.069\ -0.013]^T$ mm and the fixed rotation angles are [-0.06 -0.03 0.018] degrees. This corresponds to a translation error magnitude of 0.072 mm, which is less than 2% of the resolution of the surface meshes (5.9mm) being registered and an angular error magnitude of 0.07 degrees. The accuracy of the registration demonstrates the usefulness of our algorithm in medical registration procedures where the transformation between data sets is unknown or highly uncertain.

Figure 3-12 demonstrates the registration of a fine elevation map to a coarse digital elevation map. The ability to register a local elevation map (perhaps generated by on-board sensors) to a coarse global elevation map is very useful in outdoor mobile robot navigation. It can be used to initially register a robot to a global map and also to correct for subsequent drift in global position as the robot traverses the terrain. The model data set in Figure 3-12 was generated by subsampling (10:1) a digital terrain map[1]; the scene data set was generated by cropping the same digital elevation map without subsampling the data. This result shows the usefulness of our al-

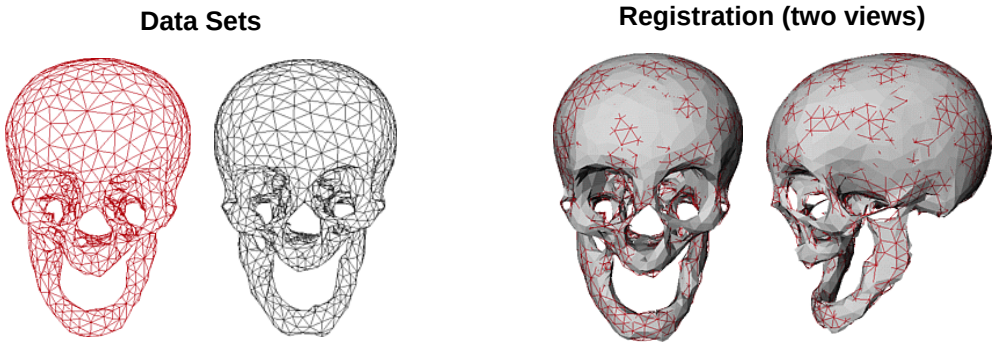


Figure 3-11: Registration of two skull data sets generated from CT. The accuracy of the registration (0.169mm) demonstrates the usefulness of our algorithm in medical registration procedures where the transformation between data sets is unknown or highly uncertain.

[1] Real elevation data collected from the Lockheed Martin facility in Denver, Colorado.

gorithm in mobile robot navigation and shows that the algorithm works even when the resolutions of the meshes being compared are substantially different.

In order to accurately convey the dependence of our algorithm on thresholds and free variables, we have provided a table of values for each of the results shown in this chapter. In Table 3-1, the first free variable, bin size, is the size of the bins used to create spin images for the model (Section 2.3) expressed in units of model resolution. The next free variable, T_{angle}, is the support angle (threshold on maximum difference in angle between surface normals) used when creating spin images (Section 2.3). N_{sp} is the number of scene points selected randomly from the scene (Section 3.1.1); it is expressed as a percent of the total number of scene points. T_{sim} is the threshold on similarity measure, expressed as a percentage of the maximum similarity measure of all correspondences generated, used to remove unlikely correspondences (Section 3.2). D_{gc} is the geometric consistency threshold, expressed in units of mesh resolution, used to remove correspondences that are not geometrically consistent with other corre-

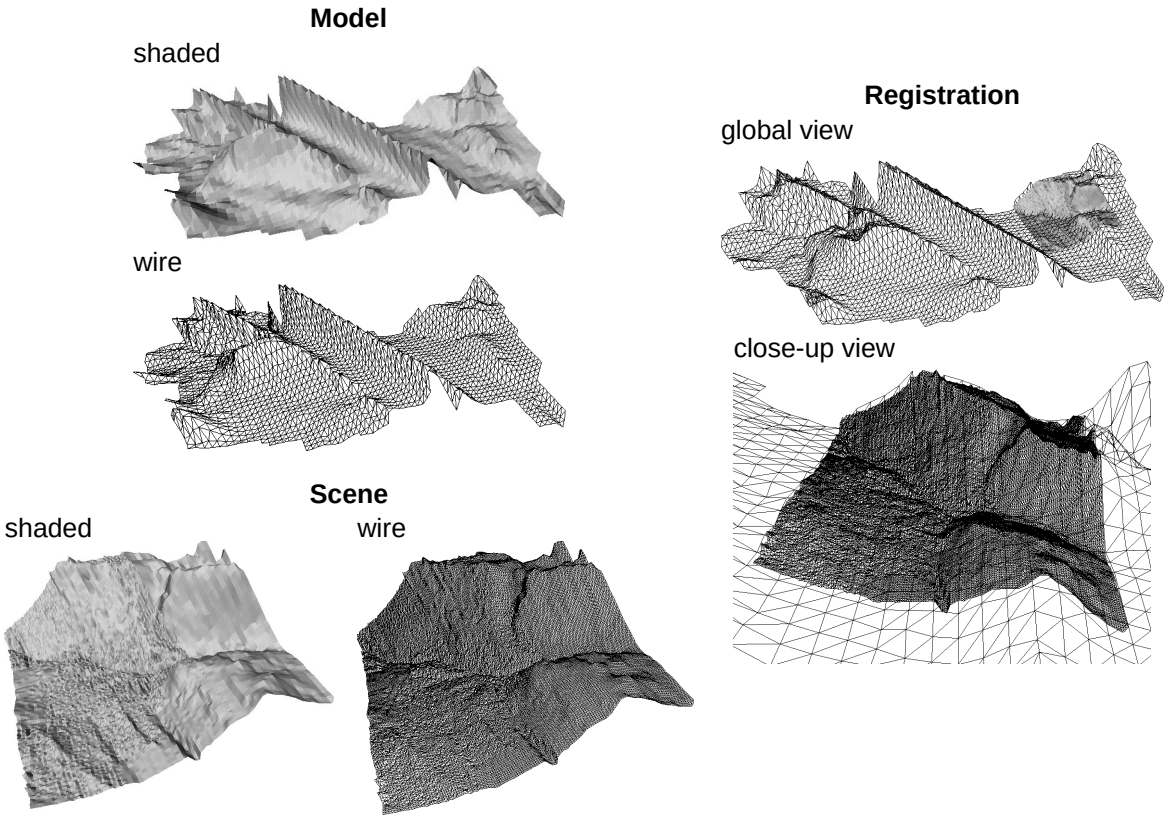


Figure 3-12: Registration of a fine local elevation map to a coarse global elevation map. This result shows the usefulness of our algorithm in mobile robot navigation and shows that the algorithm works even when the resolutions of the meshes being compared are substantially different.

spondences (Section 3.2). D_V is the maximum search distance in the verification algorithm (Section 3.4) also expressed in units of mesh resolution.

The parameters shown in the table are all basically the same. The valve and bones results involve matching a complete model do a partial scene, so the support angle is set to 60˚. For the rest of the experiments, the support angle is unconstrained (180˚) because either partial scenes are being matched to partial scenes or complete scenes are being matched to complete scenes. The elevation map parameters are different from the other results, because a very dense scene is being matched to a coarse model. In this case, less scene points need to be selected, so N_{sp} is lower (5%). Since the model resolution is much larger than the scene resolution, the bin size is set to 1 in order to have more of the variation in the scene represented in the spin-images. D_V is lowered in order to decrease the spreading of correspondences between model and scene. Sufficient correspondences for validation will be established with a low verification threshold because of the large ratio of scene points to model points.

Table 3-1: Thresholds and free variables for recognition.

result	bin size	T_{angle}	N_{sp}	T_{sim}	D_{gc}	D_V
valve (Figure 3-8)	2	60	20%	50%	0.25	2
femur & pelvis (Figure 3-10)	2	60	20%	50%	0.25	2
multi-view1 (Figure 3-9)	2	180	20%	50%	0.25	2
multi-view2 (Figure 3-9)	2	180	20%	50%	0.25	2
skull (Figure 3-11)	2	180	20%	50%	0.25	2
elevation map (Figure 3-12)	1	180	5%	50%	0.25	0.25

3.6 Discussion

Spin-images offer a method for performing 3D surface matching using an image based representation. This has allowed the application of powerful image-based tools, like image correlation, to the problem of matching spin-images and their associated oriented points. The underlying assumption in our algorithm is that spin-images from corresponding oriented points will be *similar enough* to be matched. This is the same assumption that is used in template matching and correlation based stereo, two pervasive methodologies in computer vision. Through the use of spin-images, we are bringing the success of image based techniques to 3-D

surface matching.

For two spin-images to be similar, the shape of the scene must be similar to that of the model. In the case of complete and continuous object surfaces (and hence continuous spin-images) the spin-images created for corresponding points in the model and the scene will be exactly the same. However, when the objects are represented as discrete points connected in a surface mesh, the spin-images from corresponding points will not be the same because the points on the two objects will usually be in different positions. Fortunately, the way spin-images are generated minimizes the effects of discretization. During spin-mapping, points are bilinearly assigned to the four nearest bins in the spin-image, thus smearing their position in space and making the exact positions of the points less relevant. Since the surface normal at a point is determined from the best fit plane in a neighborhood of a point, the calculated surface normals of corresponding oriented points will be similar (if the local shape does not have large curvature.) Since the connectivity of the surface mesh is used only when determining the nearest neighbors of a point for the surface normal calculation, it will also have little effect on the appearance of the spin-images.

Matching surfaces by establishing point correspondences between a model and a scene data set is not a new idea. Previous methods have relied on the extraction of a small number of feature points between which correspondences can be established [34]. Many times these features are located at the positions that are most susceptible to noise on the surface of the object (e.g., edges, corners). Our method is novel in that it considers all of the points on the surface of an object for establishment of point correspondences, so it does not have to rely on a possibly unreliable feature extraction method. Furthermore, correspondences are established between stable locations on the surface of the object, making the computed transformation from model to scene more accurate. Our approach can be considered an integral approach in that it uses all of the points available, and lets the matching algorithm decide, at run-time, which points are the best for recognition.

Because our algorithm does not construct coordinates frames from multiple points, its computational complexity is much less than that attributed to methods of basis geometric hashing. Let S be the number of points in the scene, P the percentage of scene points selected from the scene, N the number of model points and D the size of the spin-images. The time to generate the model

spin-image stack (which can be done off-line) is $O(N^2)$ because a spin-image is generated for every point on the model and each spin-image requires the spin-mapping of every point in the model. The size of the model spin-image stack is $O(ND)$ because there is one spin-image for every model point. The establishment of correspondences between the model and the scene requires the creation of scene spin-images and the comparison of scene spin-images to model spin-images. Creating scene spin-images is $O(PS^2)$ because S scene points must be spin-mapped for each of the PS spin-images selected from the scene. Comparing scene spin-images to model spin-images is $O(PSND)$ because all PS scene spin-images must be pixelwise multiplied with all of the model spin-images. Since $P \leq 1$ and N is on order of S, the total complexity for matching spin-images reduces from $O(PS^2 + PSND)$ to $O(S^2D)$.

The number of correspondences after spin-image matching C is much smaller than number of model points and scene points. Filtering correspondences is $O(ClogC)$ since our filtering procedure requires sorting. Grouping correspondences is $O(C^2)$ since the geometric consistency measure must be computed for each pair of correspondences.

The ICP verification algorithm is worst case $O(NlogN)$ for each iteration of the algorithm, assuming that all of the model points are brought into correspondence with scene points. The number of iterations in our verification algorithm is fixed independently of N, so the complexity of verification is still $O(NlogN)$.

The computational complexity of the algorithm is not prohibitive as is born out in the recognition times (real-time) shown in Table 3-2 for the results discussed in the previous section. In Table 3-2, stack create time is the off-line time taken to create the spin-image stack of the model, match time is the time used to create and match all scene spin-images to the model stack and generate transformations, and verify time is the time taken to verify the transformations generated including the refinement of the transformations using our ICP algorithm. The sum of the match time and verify times encompasses all computations done on-line for recognition. No feature extraction or segmentation stage is needed before recognition. All times are real wall clock times on a Silicon Graphics O2 with a 174 MHz R10000 processor. At this point, it should be stressed that the times shown constitute the total time needed to match surfaces; no additional time is needed to preprocess the surfaces before matching. The unusually long time taken in matching of elevation maps is due to the exceptionally large number of scene points

(S = 22447) that must be spin-mapped for each scene spin-image.

Table 3-2: Recognition timing statistics.

result	# model points (N)	total # scene points (S)	% scene point selected (P)	spin image size (D)	stack size in bytes	stack create time	match time	verify time
valve (Figure 3-8)	2433	4273	20%	200	1.9M	79s	60s	8s
femur & pelvis (Figure 3-10)	2313	2792	20%	200	2.2M	41s	40s	21s
multi-view1 (Figure 3-9)	2716	2349	20%	200	1.9M	101s	27s	22s
multi-view2 (Figure 3-9)	2369	2349	20%	200	1.8M	78s	23s	23s
skull (Figure 3-11)	1541	1532	20%	200	1.2M	59s	7s	2s
elevation map (Figure 3-12)	2216	22447	5%	200	1.8M	75s	538s	44s

3.7 Related Work

A comprehensive overview of model representation in 3-D computer vision is given in [37] and comprehensive overviews of methods for establishing correspondences and verifying matches are given in [34][36]. Because of their direct relevance to our algorithm, some review of work in the areas of geometric hashing, structural indexing and the ICP algorithms is necessary.

The term geometric hashing was first coined by Lamdan and Wolfson [59][60] to describe their work utilizing a look up table to speed the establishment of correspondences in recognition. Geometric hashing can be broken into two categories: basis geometric hashing and structural indexing or curve based geometric hashing [86]. In basis geometric hashing, local features are extracted from the model of an object and basis coordinate systems are generated using tuples of these features. Geometric constraints are encoded in a lookup table by computing the coordinates of other model features with respect to the current basis, and storing an identifier to the current basis at the location of the coordinates in the lookup table. This preprocessing stage is done for all possible tuples of features on a model and all models in the model library. At recognition time, features are extracted from the scene and a basis from a tuple of features is generated. The likelihood of this tuple belonging to a particular model is determined when the other scene features vote into the hash table. Model tuples receiving a large number of votes in the table are possible matches to the scene tuple. The reliance on tuples of features makes the combinatorics of basis geometric hashing prohibitive when dealing with large numbers of fea-

tures.

Structural indexing or curve geometric hashing does not use tuples of features like basis geometric hashing, but rather computes local signatures on objects that are stored in a look up table. Usually these signatures are curves that are processed in some way for easy comparison [86]. In the preprocessing stage, signatures are computed for each model and then stored; at recognition time signatures are computed in the scene and compared to the model signatures to establish correspondences. Structural indexing is an excellent way to establish correspondences between arbitrarily shaped objects because it makes no assumptions about the shape of objects.

Possibly the work most relevant to our work in establishing correspondences is that of Guéziec and Ayache [36] for the matching of 3-D curves. They establish correspondence between two curves using an indexing scheme that stores all curve points in a lookup table. Their method requires the extraction of 3-D curves (features) from volume data and then the calculation of sensitive Frenet frames on the curves. Chua and Jarvis [14] present a method that is similar to ours for recognition of free-form objects using point correspondences. However, their method establishes correspondences by comparing principal curvatures of points, and therefore requires the calculation of sensitive Darboux frames at every point on the object. Our work requires no curve extraction and depends only on relatively stable surface normals for calculation of coordinate bases.

Our algorithm for establishing correspondences between arbitrarily shaped models and scenes is a combination of the basis geometric hashing and structural indexing. Local bases are computed at each point on the surface of an object using the coordinates of the point and its surface normal; the coordinates of the other points on the surface of the object are then used to create a descriptive spin-image for the point. Information from the entire surface of the object is used to generate spin-images, instead of a curve or surface patch in the vicinity of the point; thus, spin-images are more discriminating than the curves used to date in structural indexing. Finally, because bases are computed from single points, our method does not suffer from the combinatorial explosion present in basis geometric hashing as the amount of data is increased.

The ICP algorithm developed concurrently by Besl & McKay [3] and Zhang [100] has been

shown to be an excellent method for registration of free form surfaces. The obstacle to using ICP for object recognition is that it requires an initial estimate of the transformation between model and scene because it is a greedy algorithm that converges to the nearest local minimum in transformation space. Another problem is handling outliers when one data set is not a proper subset of the other. Our ICP verification algorithm avoids local minimums in pose space and handles the outlier problem by boot-strapping an ICP algorithm with previously established point correspondences.

In the next chapter, we present an object modeling algorithm based on surface matching. In Chapter 5, we discuss how surface matching can be used for multi-model object recognition in cluttered scenes.

Chapter 4
Application: Object Modeling

There is an increasing interest in modeling scenes and objects for virtual reality applications, in the areas of business (real estate, architecture, information-dispensing kiosks,) education (electronic museums and multimedia books), entertainment (interactive 3-D games, movies) and reverse engineering. The option of creating virtual reality models by capturing real scenes through video cameras or range sensors is receiving particular attention given the labor-intensive, and thus expensive, nature of creating models by hand using a 3-D geometric modeler. The problem with creating a model of a complex object is that a single view cannot completely convey the shape of the object; thus, merging of multiple views acquired at different locations is usually necessary. In general, merging of multiple views is a two step process: first, the views are registered then they are integrated into a seamless 3-D model.

In many modeling systems, multiple views of the object are obtained by positioning the object with a calibrated turntable or robot arm [16][97]. By reading the encoders of the object positioning system, the transformation between views of the object is determined precisely. In these systems, the registration component of object modeling is solved through expensive hardware. Unfortunately, there are many situations where precisely positioning an object is not possible; the object may be too large, or the object may be attached to another object. Furthermore, reliance on an object positioning system prevents the modeling of multi-object scenes. If view registration can be performed automatically in software, then single object and multi-object scenes of any size can be modeled without using an expensive object positioning system.

As we showed in the previous chapter, surface matching with spin-images can be used to automatically align two views of an object. Based on this, we have developed an object modeling

system that creates complete models of objects from range images. Besides building models for object recognition, the purpose of our system is to demonstrate the effectiveness of spin-images for matching surfaces of many different shapes, without knowledge of the transformation between surfaces. Using our system, we have built twenty models of objects with a wide variety of shapes, without an expensive object positioning system.

Object models are created as follows. First, between ten and twenty views of the object are taken by a structured light range camera. When taking views, the object is placed by hand in front of the sensor, so the transformations between views are unknown. Next the range views are processed and turned into surface meshes. A multi-step procedure that combines registration and integration is then applied to the surfaces to place them in a single coordinate system. The procedure is designed to limit accumulation of errors that can occur due to concatenation of transformations. Finally, the surfaces are integrated into a single surface and texture mapped with the appearance that is available from the range camera.

In the following sections, we first describe the range image acquisition system and processing of range views for matching and integration. Next, we briefly describe the range image integration algorithm used in object modeling because it is an important component of the multi-step modeling procedure. Next, we explain the role of surface matching in registration of range images that makes object modeling, without a model positioning system, possible. We conclude with a description of the complete object modeling system and a presentation of results.

4.1 Data Acquisition

In our system, object surface data is acquired using a commercial K^2T structured light range camera which consists of a structured light projector, a black and white camera, a calibration cube and a computer workstation with a video digitizing board. Absent from our system is a method for controlling the pose of the object for view acquisition. Range images are created as follows: The structured light projector projects a sequence of horizontal line patterns into the viewing frustum of the camera. The patterns binary encode the volume in front of the projector; each binary code corresponds to a specific plane of light emitted from the projector. During image acquisition, the camera assigns a binary code to each pixel in the image based on which patterns of light change the intensity of the pixel from dark to light. The range of the pixel is

then calculated through triangulation between the pixel viewing ray and the plane corresponding to the binary code of the pixel. Computing the range for every pixel results in a range image. Co-registered intensity is available from the range camera by storing the intensity of the image acquired when the entire viewing area is illuminated by the projector.Figure 4-1 shows the data acquisition set up and the resulting data for a single view of a rubber duckie object.

The position of the camera with respect to the projector must be determined so that accurate range values are generated; calibration of the system is performed by imaging the calibration cube. On all sides of the cube are white dots in known 3-D locations with respect to the cube coordinate system. The calibration cube is imaged by the camera, and the user interactively locates the image positions of white dots from three sides of the cube. The binary code for each white dots is computed by projecting the patterns of light onto the calibration cube. Given the binary codes for the points, their image position and the 3-D coordinates with respect to the cube coordinate system, the location of the camera and projector with respect the coordinate system of the cube is determined through a non-linear minimization (for example, see [27]).

In our object modeling system a single view is acquired by manually placing an object on a matte black background in front of the scanner. A threshold in intensity is interactively set to remove the background from the range image and the range image is acquired. The range image is converted into a surface mesh by assigning a vertex to each range pixel in the foreground of the image. The connectivity of the surface mesh is created by computing the Delaunay trian-

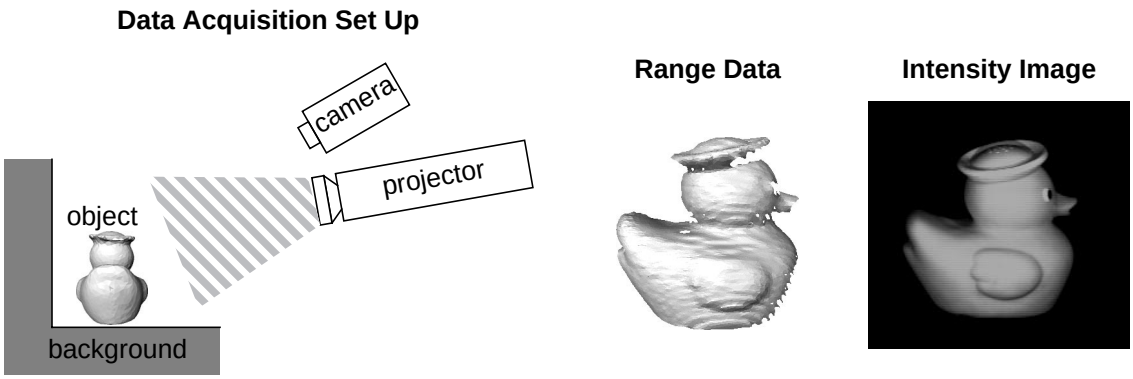


Figure 4-1: Data acquisition setup. A structured light projector generates planes of light which are imaged by the camera. Each plane of light creates a contour of bright pixels in the camera image whose shape is defined by the shape of the object. The 3-D location of the contour pixels are determined by intersecting the viewing ray of the pixel with the equation of the plane of light generating the contour. The result of data acquisition is a set of 3-D points, a camera projection matrix and a intensity image.

gulation of the image coordinates of the range pixels. The Delaunay triangulation is used in order to fill holes in the object surface that were assigned to the background because of low intensity. Next, edges that are longer than two times the resolution of the mesh are removed from the mesh because these are most likely caused by outlier or mixed pixels. Edge removal may create isolated vertices, so the mesh is processed to remove isolated vertices and dangling edges. Once a clean mesh is created, it is smoothed using a smoothing without shrinking filter [88]. The final stage in mesh processing is to simplify the mesh using the algorithm in Appendix A to two different levels: a coarse level for surface matching (coarse-mesh) and a fine level for registration after surface matching (fine-mesh).

To create the complete object model, multiple views of the object are taken and processed. Each view is taken by manually placing the object in front of the sensor, so that no knowledge of the transformation between views is available. The views are taken somewhat systematically though. Adjacent views are taken about 45 degrees apart so that sufficient overlap exists for surface matching. Furthermore, the qualitative position of the views is stored (i.e., view 2 was taken next to view 1) so that knowledge about which surfaces to match is available. The views are selected by the user so that most, if no the entire surface, is covered.For a typical model, between 12 and 20 views are taken depending on the complexity of the object. This number of views is between two and ten times less than the number of views taken in typical view integration algorithms, so image acquisition time is shorter.

Currently, the user inputs the adjacency of views into the object modeling system. However, surface matching can automatically decide which views are next to each other as follows. Given a particular view, match it to all of the other views taken. If the overlap after surface matching is large, then the two views are adjacent. We did not pursue automatic detection of adjacent views because it is fundamentally quadratic in the number of views taken, and the adjacency information was available from the user. In the next section, we describe the surface integration algorithm we use to merge multiple surfaces.

4.2 Surface Integration

The purpose of integration is to combine separate 3-D surfaces into a single seamless surface. Integration assumes that the 3-D surfaces have been registered prior to integration. Registration

can be provided by a calibrated image acquisition system or, as shown in the next section, by automatically registering the surfaces using surface matching. This section describes different techniques for surface integration and then briefly describes the surface integration algorithm used for object modeling; the details of the algorithm can be found in Johnson and Kang [53].

4.2.1 Related Work

The integration problem is an active area of research where the common approaches are divided into two groups based on the type of data input into the algorithm. The first group integrates unstructured point sets. The second group of algorithms is supplied structured data which provides some knowledge, usually in the form of a surface mesh, about the underlying surface shape. The structured data approaches can be broken down further into surface-based and volumetric approaches.

Integration algorithms that can be applied to unstructured point sets are useful when no underlying surface information is available. The surface is constructed using proximity information in 3-D space. Boissonnat [10] developed an algorithm for efficient computation of the Delaunay tetrahedralization of space. Veltkamp [96] creates surfaces from unorganized points by generalizing the concept of closest point using a γ-neighborhood graph, when constructing a 3-D tetrahedralization of space. Hoppe et. al. [43] use an augmented Euclidean Minimal Spanning Tree to create a signed distance function from a set of unorganized points. They then polygonize the signed distance function using the Marching Cubes surface polygonizer. Bajaj et. al. [2] use alpha-shapes and Bernstein-Bezier forms to construct smooth surfaces from a set of unorganized points. Because unstructured point algorithms have no topological information to begin with, they produce smooth surfaces which can give unreliable surface estimates near discontinuities in the scene.

The second group of algorithms assumes that some information describing the topology of the surface to be reconstructed is available. Usually this information is conveyed by connectivity information obtained through the data acquisition process (e.g., scanning). With connectivity, the surface normal at each point can be calculated, giving a richer description of the shape of the object than 3-D points without surface normals.

Surface-based algorithms for integration of structured points usually operate on polygonal

meshes. Soucy and Laurendeau [83] partition the points into disjoint sets based on a Venn diagram of views. Within each disjoint set they create a rectangular grid of surface points which are integrated along boundaries in the Venn diagram. Turk and Levoy [93] developed a method which zips together overlapping surface meshes followed by adjustment of mesh vertex positions based on all the overlapping data. The algorithm of Rutishauser, et al. [74] uses a sensor error model to combine redundant points followed by a retriangulation step. By using the surface information, these algorithms will produce better results than those produced by the unorganized point algorithms. However, dependence on a view based retriangulation step will result in poor results near complex regions of high curvature. Chen and Medioni [13] avoid the view dependent retriangulation step by growing a deformable surface to the surface data. However, their approach assumes the object being modeled is genus zero which is not true when modeling complex scenes.

The final group of integration algorithms constructs a continuous 3-D implicit function describing the surface using a volumetric data structure to discretely sample the function. Once the implicit surface is constructed, it is polygonized using a the Marching Cubes algorithm to create the surface from the volumetric data. The methods vary in how the implicit surface is constructed and the volumetric data is organized. Hilton, et al. [41] and Curless and Levoy [16] have developed volumetric integration algorithms that construct a weighted signed distance function to the surface from structured point data. Hilton et al. use surface normal and distance to compute the signed distance function. Curless and Levoy augment their algorithm with a space carving step to clean up the meshes produced by polygonization. Wheeler [97]has developed a robust volumetric approach to surface integration based on a signed distance function. His algorithm is able to find the best consensus surface given multiple noisy surface estimates.

4.2.2 Volumetric Occupancy Grid Integration

The algorithm we used for surface integration [53] was developed at Digital Equipment Corporation’s Cambridge Research Lab. This algorithm is similar to the volumetric approaches to integration that construct a 3-D implicit surface function. However, the construction of the implicit surface is novel in that probabilistic occupancy grids and robust ridge detection are used to recover composite 3-D surfaces. Occupancy grids [25][62] represent the likelihood of 3-D spatial occupancy through voting. This representation is very attractive for integrating surfaces

because it is incremental, it is simple, it allows for free-form objects, and it is flexible in allowing the incorporation of data from different sensors and their models.

The first step in integration is to build an occupancy grid representation of the surfaces being integrated. The occupancy grid is data structure composed of voxels. Each voxel contains a scalar measure of the probability of surface and a likely surface normal direction at that position in the volume. Memory is conserved in the occupancy grid by allocating only voxels that are near surfaces being integrated; the voxels are efficiently indexed by storage of raster order index of each voxel in a a binary tree.

Surface evidence is accumulated in the occupancy grid by voting into the grid using a sensor model. During integration, the probability of surface around the 3-D position of each vertex is updated using the sensor model. After accumulation of surface probability for all vertices for all surface meshes being integrated, the occupancy grid contains an intensity function corresponding to the probability of surface. Slices though the probability of surface for 15 views of an object are shown in Figure 4-3. The particular sensor model comprises two cylindrically symmetric gaussians centered on the vertex. One accounts for sensor error, and its axis is oriented along the viewing direction. The other accounts for the discrete position of vertices in defining the surface, so it is oriented along the local surface normal. A 2-D slice of the sensor model geometry is shown in Figure 4-2.

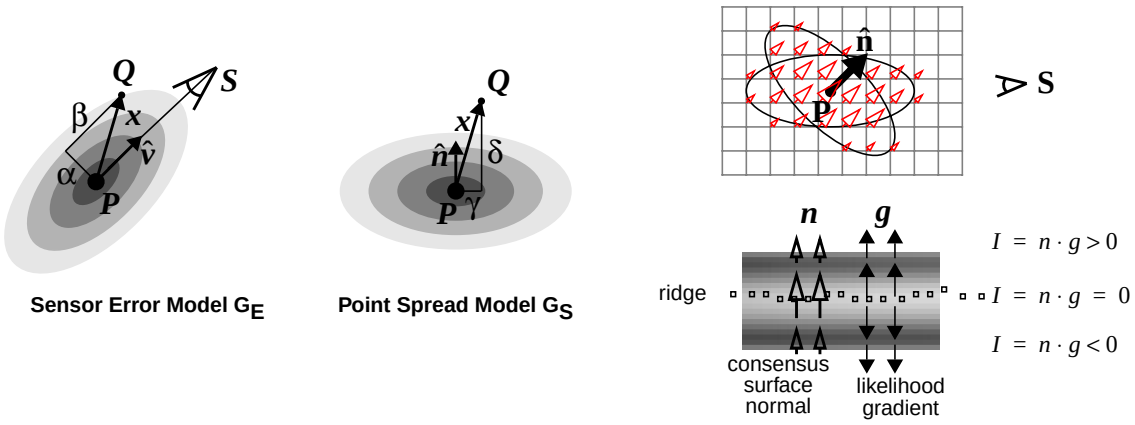


Figure 4-2: Geometry for the two components of the sensor model: the Sensor Error Model, a cylindrical gaussian oriented along the sensor viewing direction and the Point Spread Model, a cylindrical gaussian oriented along the surface normal. The occupancy grid stores consensus surface normal and is updated by each data point using vector addition (top). The consensus surface normal defines the direction along which to search for local maxima in surface likelihood. (bottom).

In addition to the probability of surface, the occupancy grid encodes likely local surface normal. Likely local surface normal in a voxel is a weighted average of surface normals from nearby vertices The weight from each vertex is equivalent to the sensor model contribution of the vertices to the voxel. The likely local surface normal in each voxel is used to robustly extract the surface from the surface probability function. First the gradient of the surface probability function is determined. Next the dot product of the surface probability gradient and the likely local surface normal is computed in each voxel. The dot product produces an implicit function describing the surface; voxels that are inside the object will have a negative dot product, and voxels on the outside of the object will have a positive dot product. Polygonization of the implicit function results in the consensus surface.

The occupancy grid can also be used to integrate the appearance of the object as seen from many views. By storing the contribution of each view to the surface probability function in each voxel, a weight for the texture from each view is generated. Once the consensus surface is made, the appearance from each contributing view is texture mapped onto the faces of the consensus surface. The end result is a 3-D surface that conveys shape as well as appearance.

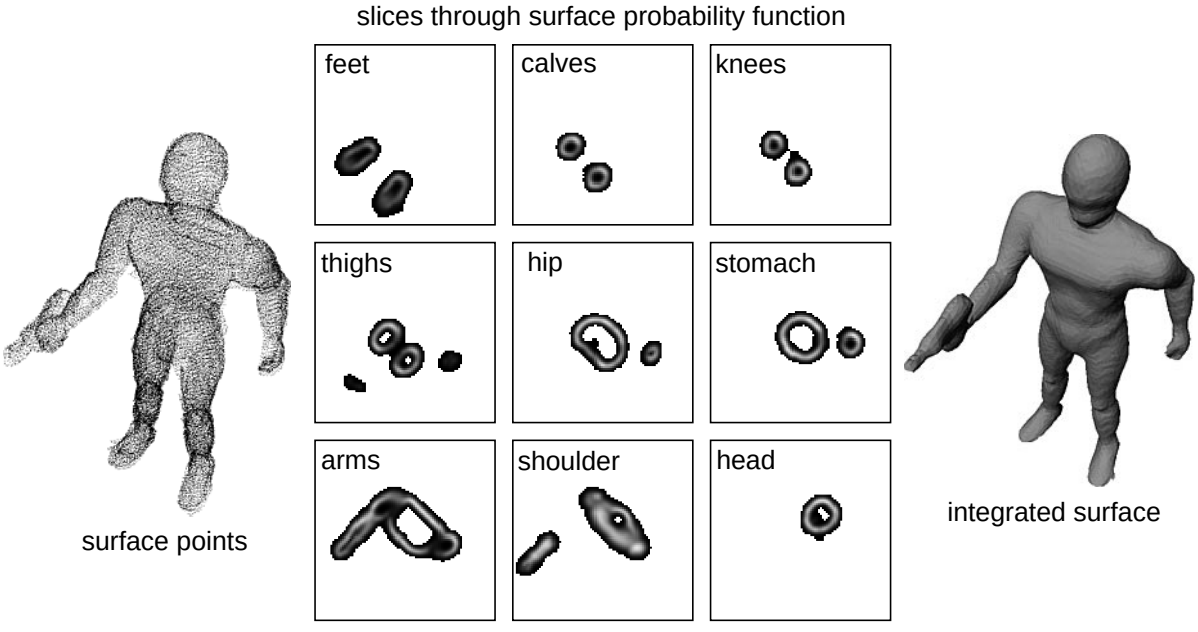


Figure 4-3: Slices through the surface probability function.

4.3 Modeling Algorithm

In Chapter 3 we presented an algorithm for surface matching. Surface matching is not the same as surface registration. Surface matching is used to find a plausible, but not necessarily the most accurate, alignment of two surfaces without knowledge of the transformation between views. Surface registration, on the other hand, is used to align surfaces precisely given an initial estimate of the transformation between views. Surface matching followed by registration will result in a very accurate alignment of surfaces with no knowledge of the transformation between views. In the following discussion, let us assume that surface registration implies surface matching followed by surface registration.

Surface registration along with surface integration can be used to build a complete model of an object from multiple views. A straight forward technique for object modeling is based on registering pairs of views. First, choose a base view to establish a coordinate system to which all other views will transformed. Next register the base view to a second view that overlaps the base view. A third view that does not overlap the base view, but does overlap the second view can be put into the coordinate system of the base view by first registering the third view to the second view. Then, by concatenating the two transformations (i.e., the transformation between the base and second views and the transformation between the second and third views) the third view can be put in the coordinate system of the base view. Repeating this process for all the views, perhaps by performing multiple concatenations, will result in all of the views being put in the coordinate system of the base view. The problem with this method is that through concatenation, any errors in the transformations from registration are compounded. This will result in seams in the registered surface on the opposite side of the object from the base view. This problem is illustrated pictorially in Figure 4-4.

In this section we describe an object modeling procedure that iteratively applies surface registration and surface integration to build up models of objects hierarchically in order to reduce the effect of accumulation of errors due to transformation concatenation. Before we describe the modeling procedure, however, we will discuss our techniques for registering two views.

4.3.1 Registering Two Views

As mentioned, surface registration is surface matching followed by precise alignment of the two views. Suppose we would like to register view *A* to view *B*. First we apply the surface matching engine to the coarse mesh representations of the two views. This will provide a coarse alignment of the two views which can be used as an initial estimate to surface registration. Depending on the registration, we use one of three different registration algorithms, all versions of the Iterative Closest Point Algorithm (ICP) [7][100].

The first ICP version, called FICP (F for Face), is similar to traditional ICP as presented by Zhang [100]. The details of this ICP algorithm are basically the same as those presented in Section 3.4 with some minor modifications to improve registration. Input into the FICP algorithm are the fine-meshes created for the two views and the transformation computed from surface matching on the coarse-meshes. A three dimensional kD-tree is computed for the vertices in fine-mesh *B*. Next, the closest vertices in fine-mesh *B* to the vertices in fine-mesh *A* are determined efficiently using the kD-tree. To improve the closest point matches, the point on the faces adjacent to the closest point in fine-mesh *B* is determined and used as the closest matching point to the corresponding vertex in fine-mesh *A* [81]. To prevent closest point matches from occurring outside the area of overlap of the two views, a threshold is placed on the maximum distance between closest points. This threshold is usually set to two times the resolution of fine-mesh *A*. FICP will give excellent registration results when the two views are already well aligned by surface matching because it establishes correspondences between vertices and faces

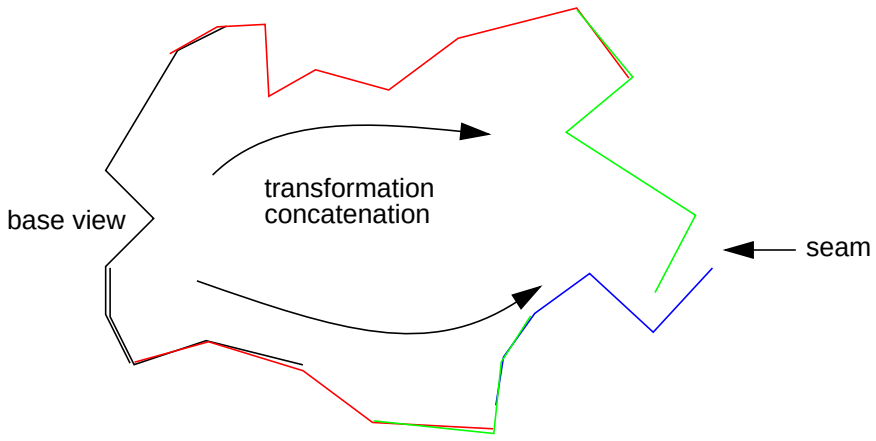


Figure 4-4: Problem with integration algorithms that require transformation concatenation. As transformations are concatenated, errors accumulate, resulting in seams in the integrated model

instead of vertices and vertices. Therefore, FICP is used in situations where the initial transformation is expected to be good.

The second form of ICP, called NICP (N for Normal), uses the vertex position as well as surface normal when searching for closest points[9][97]. As in FICP, the inputs to NICP are the fine-meshes created for the two views and the transformation computed from surface matching on the coarse-meshes. Next, a six dimensional kD-tree is computed for the vertices in fine-mesh B using vertex position and surface normal coordinates. The scale between vertex normal and position coordinates needs to be established so that meaningful six-dimensional distances can be computed. The mesh resolution gives the median distance between vertex position, so it is the scale on which differences between closest vertex positions are determined. It is appropriate to scale the vertex normals by the mesh resolution so that comparison of vertex normals are on the order of the comparison between vertex positions. Therefore, before creating the kD-tree, the vertex normals in fine-mesh B are scaled from unity by the mesh resolution. The other difference between NICP and FICP is that closest points are computed between vertices and vertices, not faces and vertices. In NICP the closest point search threshold is set to two times the mesh resolution. NICP is better than FICP at registering two surfaces when the initial transformation is poor because the addition of normals to the distance computation makes the matching between closest points more accurate. Therefore, NICP is used in situations where the initial transformation could possibly be poor.

The final form of ICP is useful when the surfaces being registered have appearance information, in the form of an image, associated with them. Color ICP (CICP) is the same as NICP except that color values associated with each vertex are used instead of vertex normals in the six-dimensional kD-tree used to compute closest points. Since the surface mesh vertices may have a greater sampling than the appearance of the surface there are some issues related to sampling that need to be addressed when using CICP. Specifically, when creating the six-dimensional kD-tree for fine-mesh B, additional vertices should be created so that all pixels in the image have an associated 3-D point. This will ensure that the shape and appearance sampling are the same. The exact details of this sampling are given in [53]. Assuming color values are normalized between on [0,1], the scale for the color values is set to the model resolution, and the closest point distance threshold is set to two times the model resolution, just as in NICP. CICP

should be used when views have appearance information and when the views are very closely registered. The views must be closely registered because the high frequency components in the appearance information will cause many registration local minima around the global minimum.

4.3.2 Modeling Procedure

To reduce the effect of accumulation of errors, we have come up with the following modeling procedure. First select two views on opposite sides of the object; call them view *A* and view *B*. Next register all views that overlap with view *A* using FICP. If appearance registration is important, follow registration with FICP by registration with CICP. Next integrate view *A* and the views adjacent to it in a single surface mesh using the algorithm described in Section 4.2. Call this consensus surface side *A*. If the views were taken about 45 degrees apart, side *A* should cover about one half of the surface area of the object. Repeat this procedure with view *B* and all of the views adjacent to it to get side *B*. Now side *A* and side *B* cover the entire surface area of the object; the next step is to stitch them together.

Select a view that overlaps side *A* and side *B*; call this view *C*. Register view *C* to side *A* using FICP. (CICP can no longer be used because side *A* will not map to a single image.) Next, register view *C* to side *B*. Concatenate the transformations from side *B* to view *C* to side *A*; this will put side *B* in the coordinate system of side *A*. Finally, register side *A* to side *B* using NICP. NICP is used because the transformation between side *A* and side *B* might be slightly inaccurate due to the transformation concatenation. If necessary, the registration between side *A* and side *B* can be refined with FICP. After registration of side *A* with side *B*, the transformation will contain fewer errors than the one created by concatenation with view *C*. The reason for this is that all of the overlapping surface area between side *A* and side *B* will be used in NICP and not just the surface that overlaps view *C*. In other words, the overlapping surface area between side *A* and side *B*, on the opposite side of the object from view *C*, will act to bring the sides together during NICP. Once side *A* and side *B* are registered, integrate them into a single consensus surface; call this surface the proto-model. Although the proto-model could be used as the final model, it may have holes, hanging surfaces or other erroneous integration artifacts. These artifacts are cleaned up in the final stage of the modeling procedure.

The final stage in the modeling procedure starts by matching all of the view surfaces to the pro

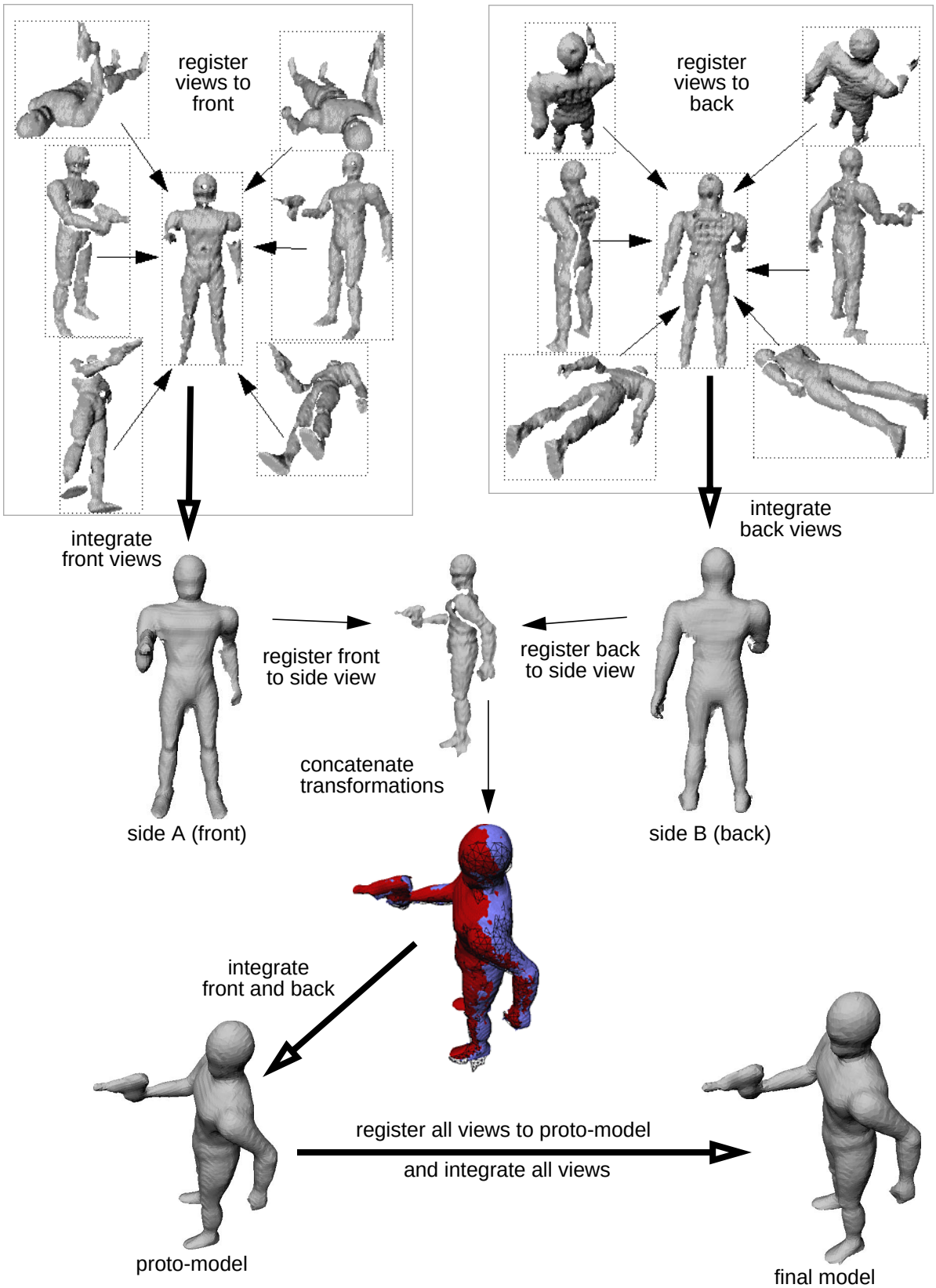


Figure 4-5: Modeling Procedure

to-model. The position of these surfaces are then refined using FICP registration. The proto-model acts as a registration aid for the individual views putting them in a single coordinate system. Once all of the views are registered to the proto-model, the views are integrated into a single consensus surface model. If appearance information is available, then it can be mapped onto the model during integration. The entire modeling procedure is shown in Figure 4-6.

This modeling procedure reduces the effect of accumulation of error by first creating one side and then the other using no concatenation of transformations. These sides are matched using a another view and concatenation, but accumulation of error due to concatenation is reduced by registering the two sides after matching. The modeling procedure is hierarchical because it registers and then integrates local sets of views, and then registers and integrates them globally. One can imagine extensions of this algorithm to modeling large complicated scenes. A tree of registration followed by integration operations proceeding from a local level to a global level could be used to create complete models with limited accumulation of errors.

Another approach to limiting accumulation of errors in multi-view registration and integration has been proposed by Bergevin and Laurendeau [6]. Their approach limits accumulation of errors by simultaneously registering multiple views of an object. Their method appears to be quite effective; however, it requires initial registrations for all of the views and a time consuming optimization procedure. Still, their algorithm could certainly be applied as a post process to our algorithm to reduce registration errors further.

4.4 Results

Figure 4-6 shows two views of a model of a toy robot. As seen in the texture mapped views of the robot, the registration is accurate enough to blend texture. The robot model demonstrates that our modeling algorithm can be used to model complicated objects; modeling this type of object with a deformable surface [13][17] would be difficult because of the narrow limbs of the robot.

Figure 4-7 shows two views of an integrated model of a PVC plumbing flow valve. This result demonstrates the ability of our algorithm to model objects of simple shape with symmetry. Figure 4-8 shows two views of a hydraulic valve. The texture of the valve is mottled because

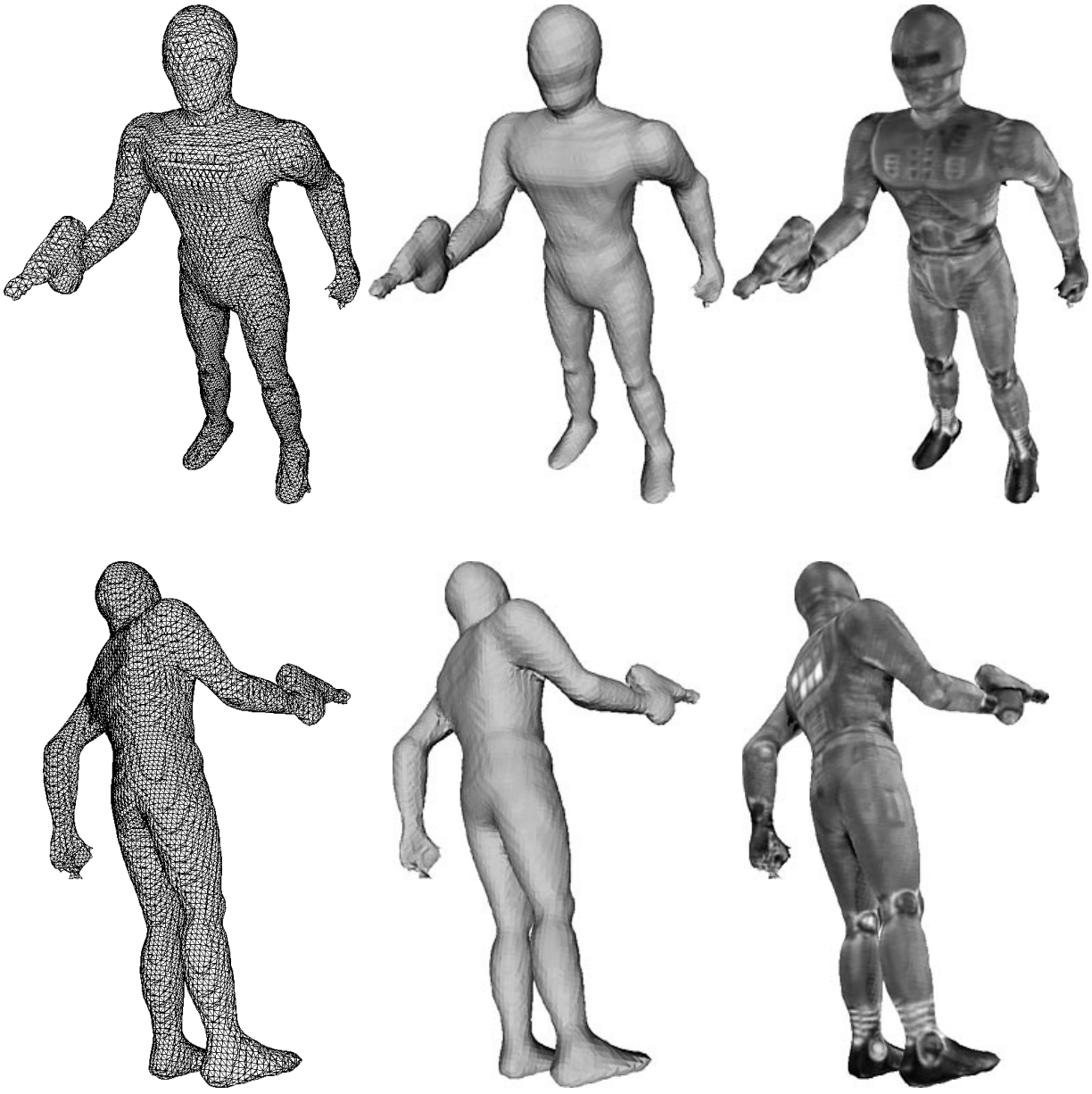


Figure 4-6: Two views of an integrated model made through registration and integration of 16 range images of a toy robot.The views are shown in three formats, surface mesh, shaded surface and texture mapped surface. The complexity of the robot model demonstrates that spin-images can be used to match complicated surfaces.

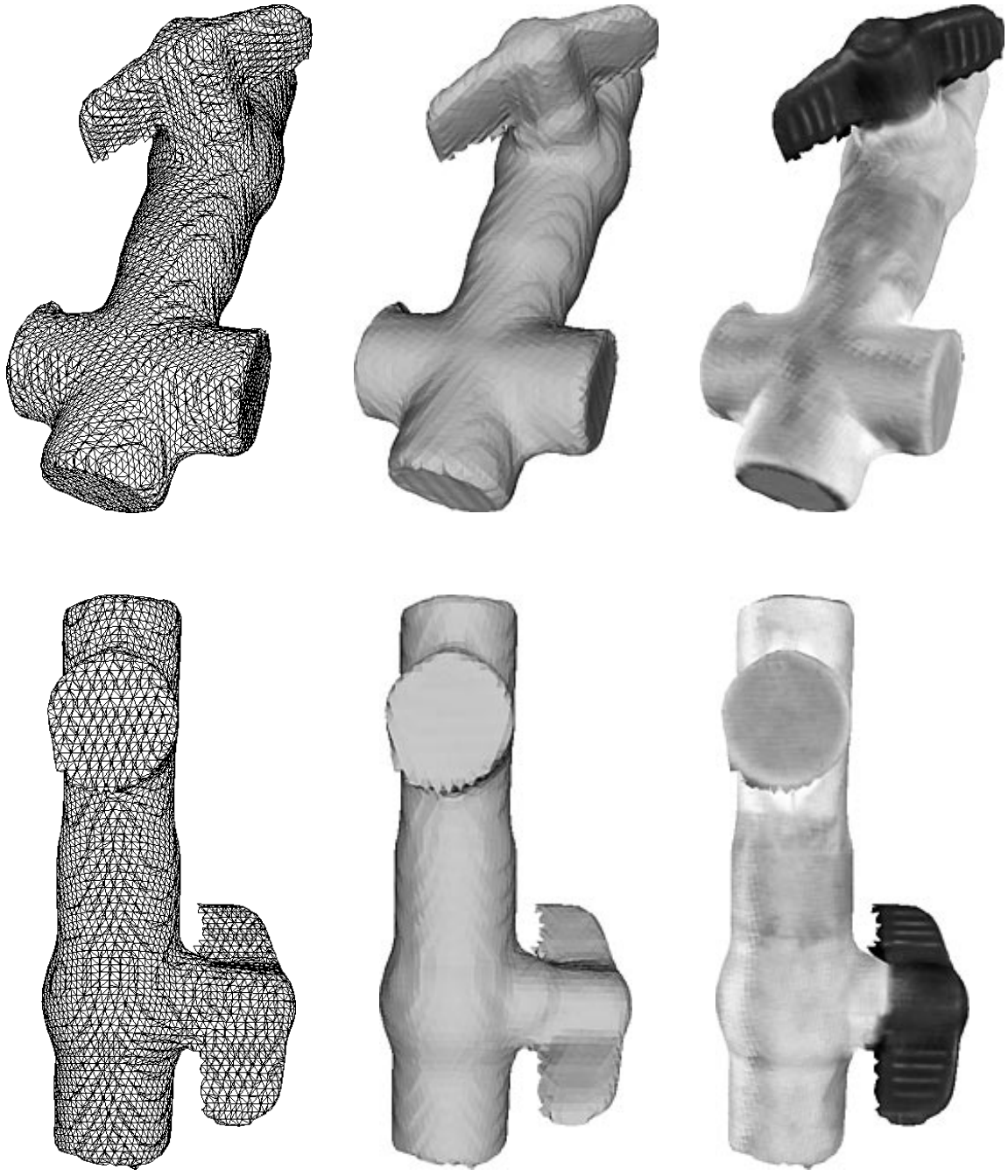


Figure 4-7: Two views of an integrated model made through registration and integration of 12 range images of a plumbing valve. The views are shown in three formats, surface mesh, shaded surface and texture mapped surface. This results shows that surface matching can be used to register simple parametric surfaces.

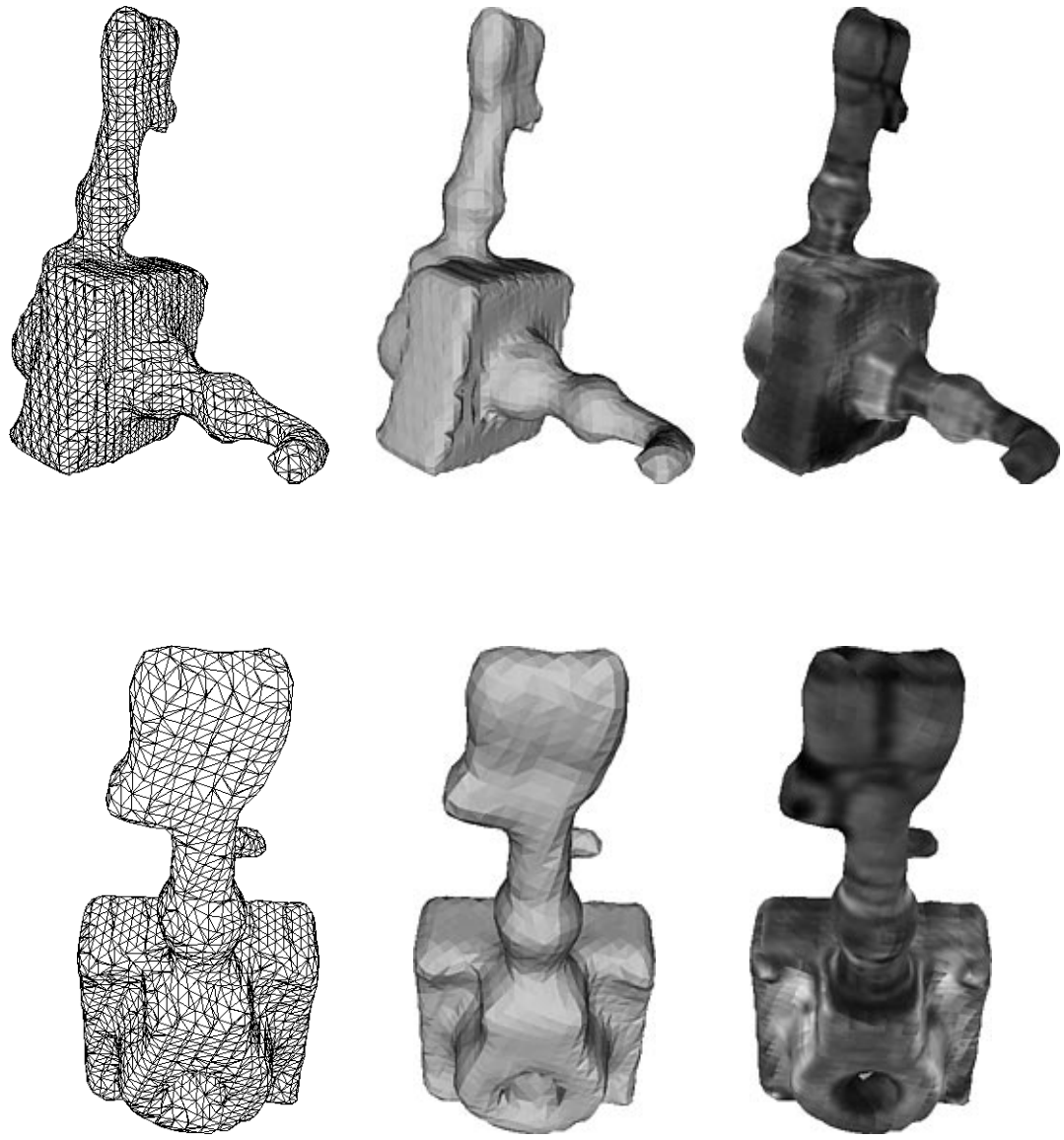


Figure 4-8: Two views of an integrated model made through registration and integration of 14 range images of a hydraulic valve. The views are shown in three formats, surface mesh, shaded surface and texture mapped surface. Because the valve object is metallic, visible specularities moved around the surface of the object as images were acquired. This resulted in the mottled appearance shown on the right because specularities were blended in with the diffuse appearance of the object.

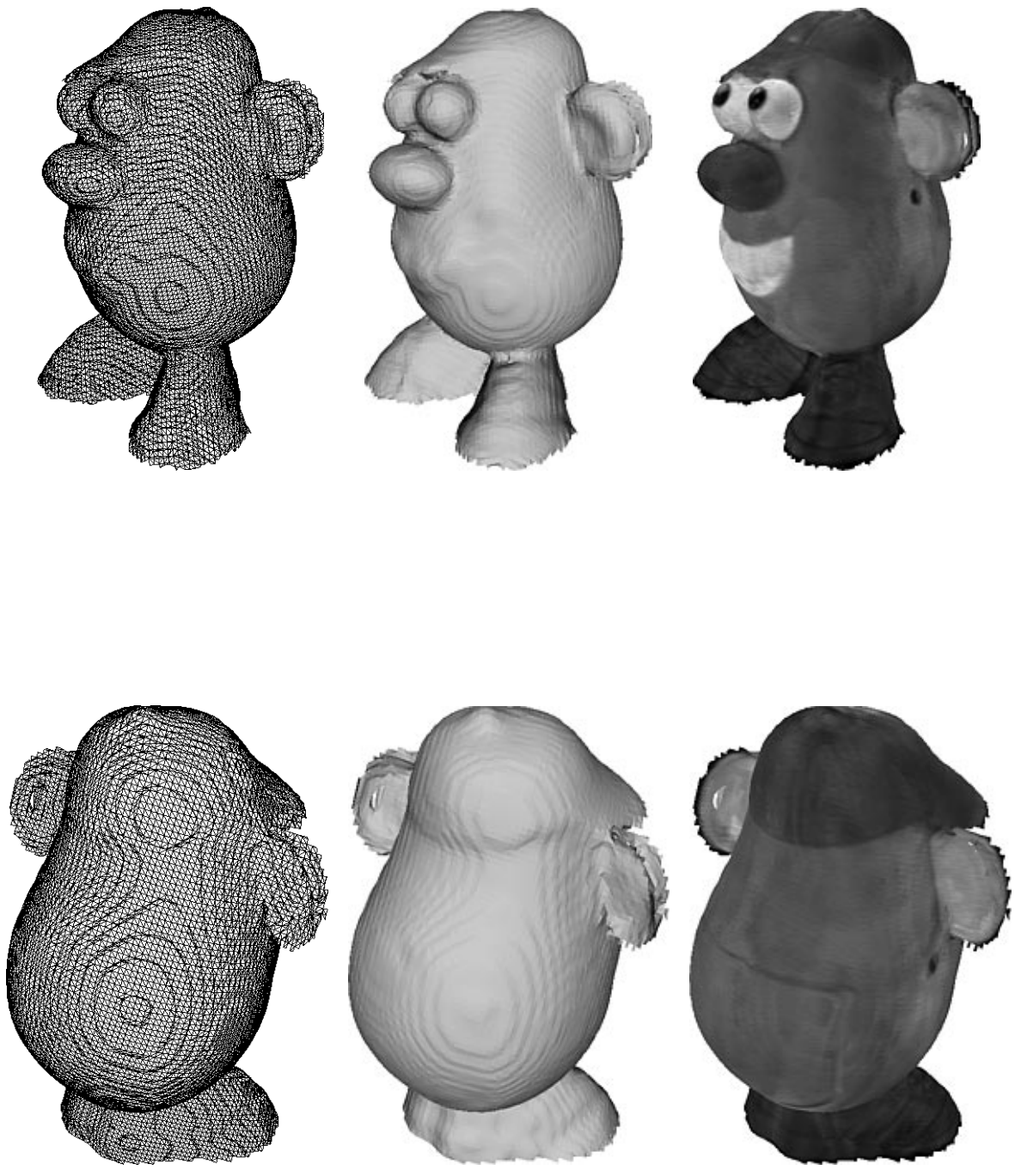


Figure 4-9: Two views of an integrated model made through registration and integration of 12 range images of a Mr. Potato Head toy. The views are shown in three formats, surface mesh, shaded surface and texture mapped surface.

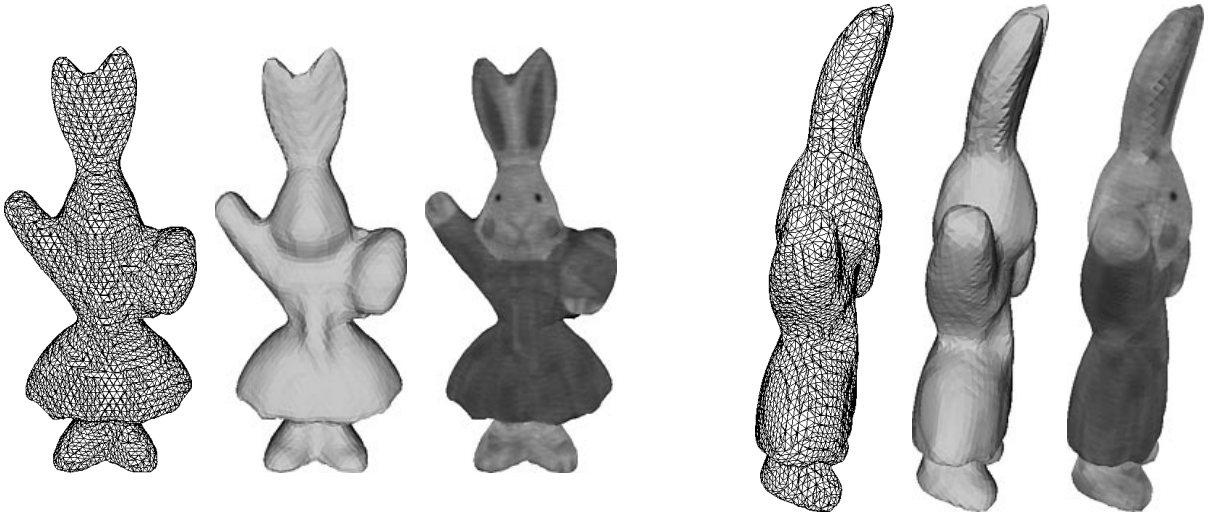


Figure 4-10: Two views of an integrated model made through registration and integration of 10 range images of a painted wooden Easter bunny. The views are shown in three formats, surface mesh, shaded surface and texture mapped surface.

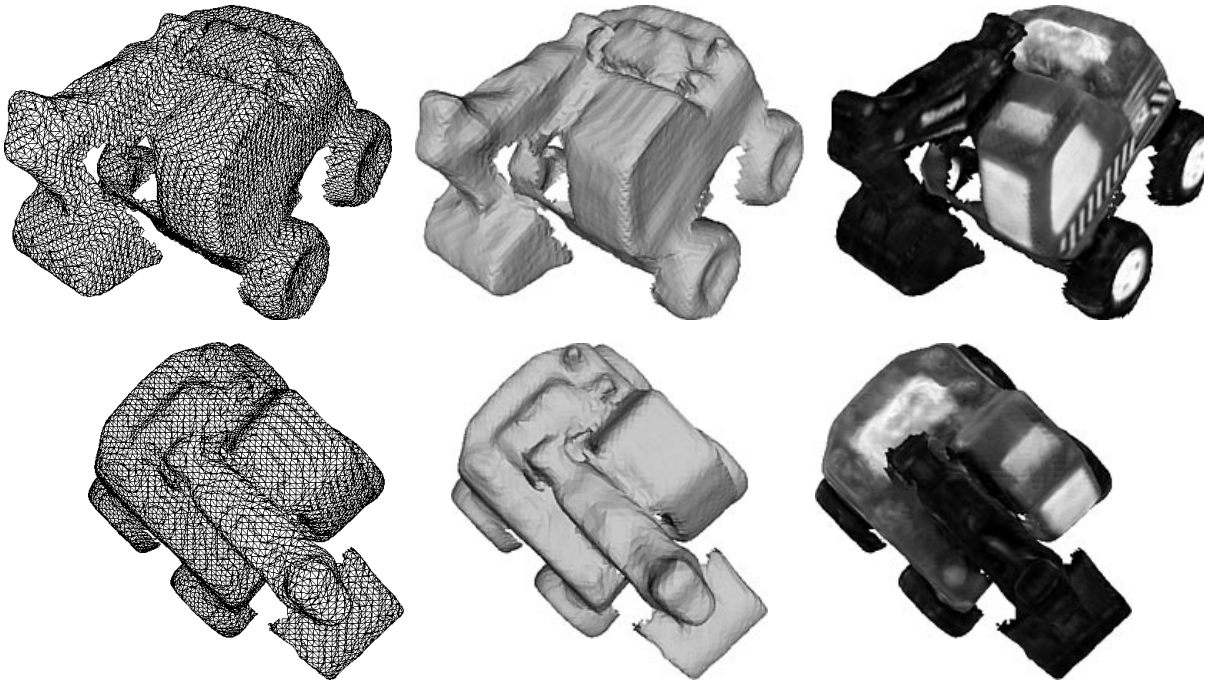


Figure 4-11: Two views of an integrated model made through registration and integration of 14 range images of a toy front end loader. The views are shown in three formats, surface mesh, shaded surface and texture mapped surface.

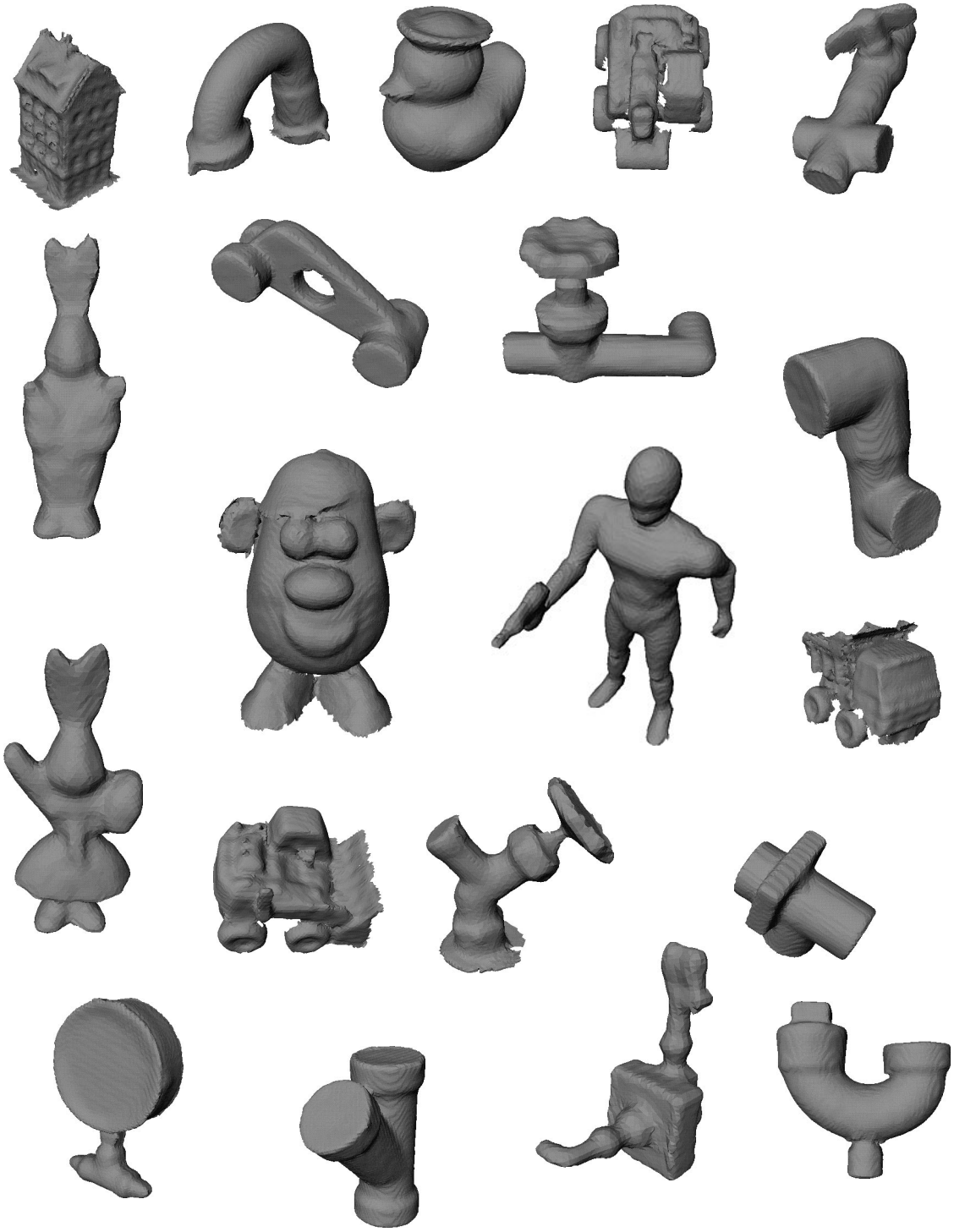


Figure 4-12: Twenty models made by modeling from range images.

the surface of the valve is metallic and hence very specular. Placing the object in different positions moves the specular highlights in different positions for different views causing the mottled appearance. To remove these view-dependent effects, a system developed by Sato [75] could be used. Figure 4-9 shows a model of a Mr. Potato Head toy and Figure 4-10 shows two views of a Easter Bunny figurine made from wood and then painted. Figure 4-11 shows the result of modeling a toy front end loader. This model is complicated and since a small number of views (14) were used to create the model, some holes exist in its surface. Figure 4-12 shows shaded views of the twenty models we created using our object modeling algorithm.

These results demonstrate that our modeling procedure can be used to generate accurate and complete models of complex objects. Furthermore, models can be built from a small number of views without any complicated and expensive object positioning system. These results also demonstrate that the surface matching engine generates accurate registration on many different complex shapes. As will be seen in the next chapters, the models generated using our modeling procedure can also be used in recognition.

4.5 Discussion

Surface matching is an integral component of object modeling because it is used to automatically align object surfaces without knowledge of the transformation between views. By building twenty models, each with over ten views, we have demonstrated surface matching between hundreds of surfaces. Matching a large number of surfaces demonstrates that our surface matching algorithm works in practice. Since the surfaces were of varying shape, it has also been demonstrated that spin-images are effective at modeling a wide variety of shapes.

The object modeling problem we are attempting to solve is difficult for three reasons. First, no knowledge of the transformation between views is available, so the sensed data must be used to align the surfaces. Second, the surface data is noisy. Third, a small number of views are taken to model the complete object. All of the factors interact. Noise and small overlap between views hinder accurate registration, yet accurate registration is needed reduce the effect of noise on the final model. Despite these conspiring factors, we are still able to construct realistic models of complicated objects.

We have presented a modeling procedure designed to limit the accumulation of errors when concatenating transformations between views. This procedure was sufficient for our needs, but in the future, we plan to look into more sophisticated methods for simultaneous registration of multiple views. In particular, it seems plausible that point correspondences established by matching spin-images could be used to simultaneously register multiple views using a principal component analysis with missing data (PCAMD) algorithm. Similarly, to Shum et al [80], PCAMD could be used to distribute registration errors uniformly over the views.

Chapter 5

Recognition in Cluttered Scenes

An important problem in robotics is the automatic recognition of objects. By definition, recognition is the process that associates new observations with previously stored knowledge. The purpose of object recognition is to assign a high level meaning or definition of an object to some data sensed in the environment of a robot. It is easier to reason using terse high level descriptions than copious low level data; therefore, for reasoning and planning tasks, high level descriptions are more desirable. By providing these high level descriptions, object recognition facilitates the planning of complicated robotic tasks.

If a model of the object is known a-priori it can be stored in computer memory for recognition. This form of recognition is termed model-based object recognition; it is a popular form of object recognition because it has many useful properties. Since models are known a priori, their representations can be generated and stored before recognition, substantially increasing the speed of recognition and the number of objects that can be recognized. Another advantage of the model-based approach is that recognizing a complete model from partial scene data fills in gaps in incomplete scene data. Furthermore, high level information not used in recognition (e.g., appearance, material type, importance) can be stored with a model and then associated with the scene data when a model is recognized, enhancing the description of the world.

A common form of model-based object recognition, recognition by localization [14][26][33][46][97], has the following stages. First a representation used in recognition of the model is created and stored. Next, the scene is converted to the same representation as the model so that the scene can be compared to the stored model representation. Then correspondences are established between similar components of the stored model representation and parts of the

scene. Next, a spatial transformation that transforms the model into the scene is computed to find the pose of the object in the scene. Finally, the model is compared directly to the scene data to verify that the match between model and scene is correct.

Different 3-D model-based object recognition systems vary in the way stored models are compared to processed scene data and the subsequent transformation computed. Comparison can be done between global properties of the scene (e.g., volume, surface fit parameters or moments) or spatially localized features (e.g., edges or corners). Global properties are very descriptive and can unambiguously identify and localize objects. However, global properties are difficult to compute when only a partial view of the object is available or the scene contains clutter. Local features are useful when recognizing objects in cluttered scenes where global properties are difficult to compute. Extraction of features in three dimensional object recognition usually takes the form of segmentation, edge detection or convolution of an interest operator. Unfortunately, methods of feature extraction are susceptible to noise; false features may be generated, or existing features may not be detected. Furthermore, complicated free-form objects are difficult to model using linear features (points, lines and planes), making the representation of 3-D free-form objects an open research issue.

Through adjustment of spin-image generation parameters, spin-images can be smoothly transformed from local to global representations. Since spin-images bridge the gap between local and global representations, they are well suited to object recognition in cluttered scenes; by localizing spin-images, they are made insensitive to clutter while still conveying enough global information to be effective in differentiating between models. This chapter explains the use of spin-images for object recognition in cluttered scenes using the recognition by localization paradigm. First, we explain our object recognition algorithm, which recognizes objects by matching model surfaces to scene surfaces without segmentation or feature extraction. Next, we show how spin-images that are insensitive to clutter can be generated by through an appropriate choice of spin-image generation parameters. Based on geometric arguments, we then develop a clutter model that explains why matching of spin-images is insensitive to clutter. The next chapter shows many object recognition results and gives an analysis of the effectiveness of recognition in cluttered scenes.

5.1 Recognition Algorithm

The major difference between surface registration and object recognition is that, in object recognition, multiple models are compared to a single scene to determine which models, if any, exist in the scene. Our object recognition algorithm based on spin-images is essentially our surface matching algorithm except that multiple model surfaces are compared to a single scene surface. In model based object recognition, the models are known a priori, so they can be preprocessed and stored in a *model library* to speed up recognition. In our case model spin-images can be precomputed to increase recognition speed. At recognition time, stored model spin-images are compared to scene spin-images. Since the model and scene spin-images encode global shape information, the scene spin-image will be most similar to the spin-image from the corresponding model point on the corresponding model. After spin-image matching, if there exist many geometrically consistent matches between the scene and points on a specific model, then that model is likely to exist in the scene. A decision as to the existence of the model in the scene can be made based on the results of surface matching verification. The details of our recognition algorithm are as follows.

First the surface mesh representations of the models in the model library must be created. We generate model surface meshes using one of two methods. Model meshes can be generated from merging of multiple range views as described in Chapter 4 or through finite element tessellation of CAD models as described in Chapter 8. Once a model mesh is created, the spin-images for the oriented points on the model surface mesh are created and stored in a model spin-image stack. Precautions, outlined in the next section, are taken during spin-image generation to ensure that model spin-images will be insensitive to scene clutter. The resolution of the meshes from different models can be different as long as the spin-images generation parameters for a model are adjusted appropriately. Furthermore, spin-images from different models can be different sizes to account for larger or smaller objects. Once all of the models are processed, the spin-image stacks for all of the models in the model library are stored for recognition. Because the model spin-images can be precomputed, recognizing objects takes less time than surface registration where model and scene spin-images have to be computed at run time.

Object recognition then proceeds in much the same way as surface matching. The major difference is that the scene surface mesh is compared to multiple model surface meshes. First,

scene points are randomly selected as explained in Section 3.1.1. Then, for each scene point, the scene spin-image is compared with all of the model spin-images for all of the models in the model library. If the different models have different spin-image generation parameters, then new scene spin-images will have to be generated for each model, based on the current model's parameters. Comparing the scene spin-image to all model spin-images is an expensive operation; fortunately, as shown in Chapter 6, there exist techniques for dramatically accelerating spin-image matching.

Given the comparisons between the scene and model spin-images, the similarity measure histogram, containing all scene to model spin-image comparisons (from all models), is computed and outliers are detected as in Section 3.1.2. The global nature of spin-images make them descriptive enough to differentiate between models (as will be shown in Chapter 7), so the outli-

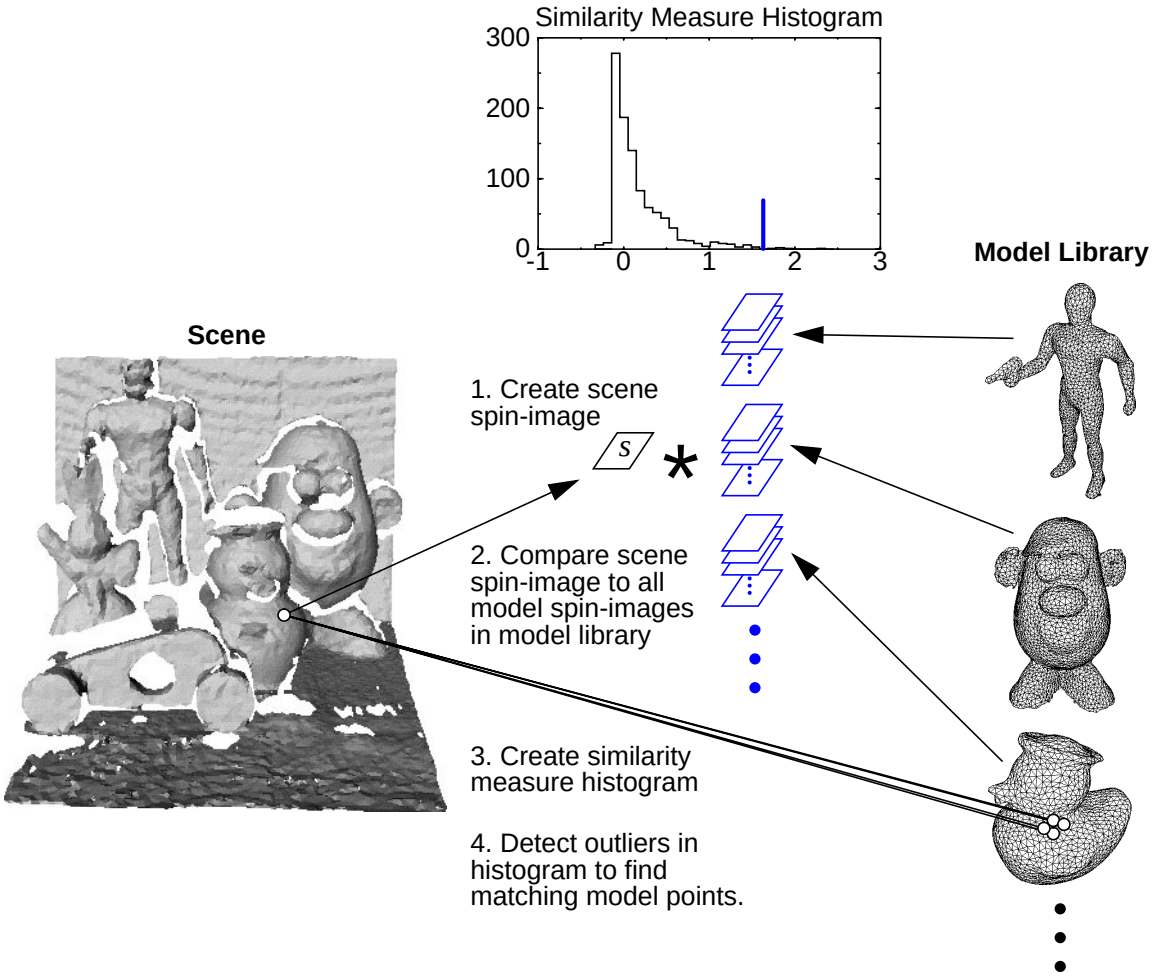


Figure 5-1: Schematic for matching a single spin-image to multiple model surfaces.

ers in the similarity measure histogram will be points on the particular model that is associated with the scene point and will not be points that come from some other model. A schematic for matching a single scene spin-image to all of the spin-images in a model library is given in Figure 5-1. Once the outliers for every scene point have been computed, the correspondences are filtered as in Section 3.2. The correspondences are then partitioned into disjoint groups, each group containing correspondences to a specific model. From each group, geometrically consistent correspondences and verified transformations are then computed using the matching engine as explained in Chapter 3.

Once a match of model to scene has gone through the verification procedure, a decision needs to be made as to the existence of the model in the scene. Verification reports the total number of correspondences between the model and the scene. At a fixed resolution, this number can vary based on the size (number of vertices) of the model, so a model-specific metric for deciding on model presence must be used. We use percentage of model surface area matched to the scene as our recognition metric. The model surface area matched to the scene is measured as the sum of the surface area from all of the model faces that contain vertices that are matched to the scene data during verification. If the matched surface area divided by the total surface area of the model is greater than a threshold, then it is decided that the model is present in the scene. We set this surface area threshold to 10%. This conservative threshold is used to prevent incorrect matches with the scene. In most robotic applications, it is better to report no matches, which is usually interpreted as no information, rather than incorrect matches which present false information.

5.2 Making Spin-Images Robust to Clutter and Occlusion

Clutter is scene data that does not belong to the model(s) being searched for in the scene. Occlusion is the absence of parts of a model surface caused by acquiring scene data from a single viewpoint. An object can occlude its backside (self occlusion), or one object can block the view of another. Clutter and occlusion make object recognition much more difficult because they add incorrect data and take away useful data. If our recognition algorithm is going to be useful in realistic setting, it must have some form of robustness to clutter and occlusion built into it. Since spin-image matching is used to recognize objects without segmentation of the scene data,

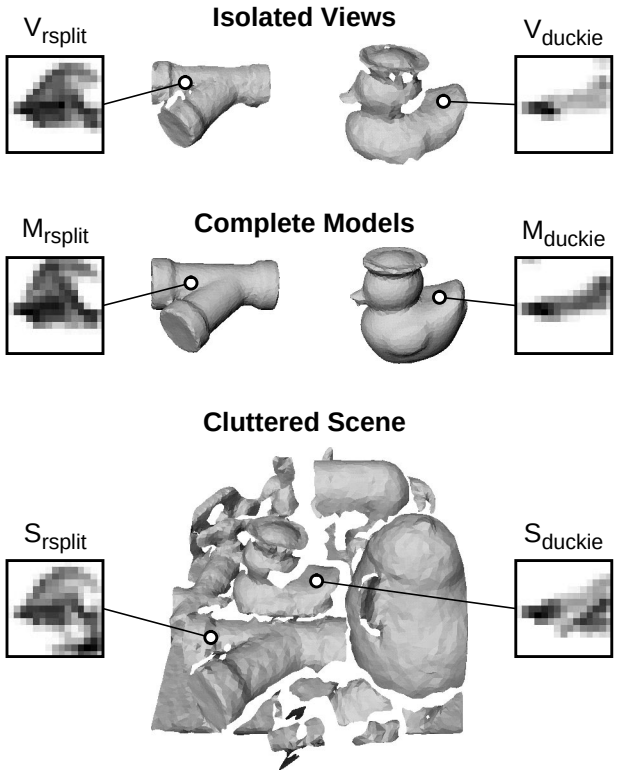
spin-image matching must be resistant to clutter and occlusion. In other words, the result of matching spin-images in the presence of clutter and occlusion should be comparable to matching in isolated scenes.

By decreasing spin-image support, spin-images are made insensitive to the clutter and occlusion that exist in real scenes. Support angle is used to limit the effect of partial views when matching a complete model to a partial scene. Support distance, controlled by setting image width, is used to limit the effect of clutter on matching by localizing the support of a spin-image. Figure 5-2 shows an experimental validation that spin-image matching is resistant to clutter and occlusion while still being effective for recognition. First a spin-image generated from a complete model of a duckie is compared to an isolated view of the duckie; a high correlation exists. However, when the duckie model spin-image is compared to a spin-image from an isolated view of an rsplit pipe, the correlation is low. Therefore, the duckie matches a view of itself better than a view of a different model. The conclusion: spin-images are discriminating enough to differentiate between different, isolated objects. As Figure 5-2 shows, this relationship is upheld when the duckie model spin-image is compared to spin-images from a cluttered scene. The duckie model spin-image matches the spin-image from the duckie surface in the cluttered scene much better than the rsplit pipe surface in the cluttered scene. This suggests that spin-images can be used for recognition in cluttered scenes. Similar reasoning using a complete model of the rsplit pipe results in the same conclusions.

5.2.1 Handling Partial Views

Partial scene views are common in 3-D object recognition because many geometric sensors return a range image which contains data from only one side of the objects in the scene. In addition, most models in 3-D object recognition are complete, so the problem of matching a complete model to a partial scene view is frequently encountered. Because the scene does not have all of the data that the model has, the scene spin-images will be different from the model scene images. This makes the matching of scene and model spin-images more difficult. By comparing spin-images only in areas of overlap (as explained in Section 2.4), some of the effects of partial views are eliminated. However, to further diminish the effect of partial views, a threshold on the greatest angle between surface normals is employed. We call this angle the support angle A_s of the spin-image.

Suppose there are two oriented points A and B on a complete object model with surface normals $\boldsymbol{n}_A$ and $\boldsymbol{n}_B$, respectively, Suppose that B is being spin-mapped into the spin-image of A. If an object is imaged with a range sensor, the surface normals of the visible points must be directed in the half of the view sphere centered on the viewing direction. For some B on the surface of the object, the angle between $\boldsymbol{n}_A$ and $\boldsymbol{n}_B$ will be too large for B to be visible when A is imaged. Assuming that the viewing direction is oriented along the surface normal of A (in general, this is not true), this angle will be 90˚. Using this assumption, all points that have surface normals with angles greater than 90˚ respect to $\boldsymbol{n}_A$ should be prevented from contributing to the spin-image of A because they will never be visible when A is visible. Therefore, when generating spin-images for partial view scenes, a threshold on the angle between surface normals should be employed to reduce the areas of overlap in these images where the two might not agree. This threshold, support-angle, is generally set to 90 or 60 degrees. As discussed in the next section, support angle also reduces the effect of clutter.



model-view correlation

	M_{rsplit}	M_{duckie}
V_{rsplit}	0.814	0.478
V_{duckie}	0.498	0.885

model-scene correlation

	M_{rsplit}	M_{duckie}
S_{rsplit}	0.809	0.426
S_{duckie}	0.545	0.746

Figure 5-2: Experimental validation the of robustness of spin-image matching to clutter. Comparing a complete model to an isolated view of the model results in high correlation. Comparison of the model to an isolated view of a different model results in low correlation. When the object models are imaged in a cluttered scene, this relationship still holds.

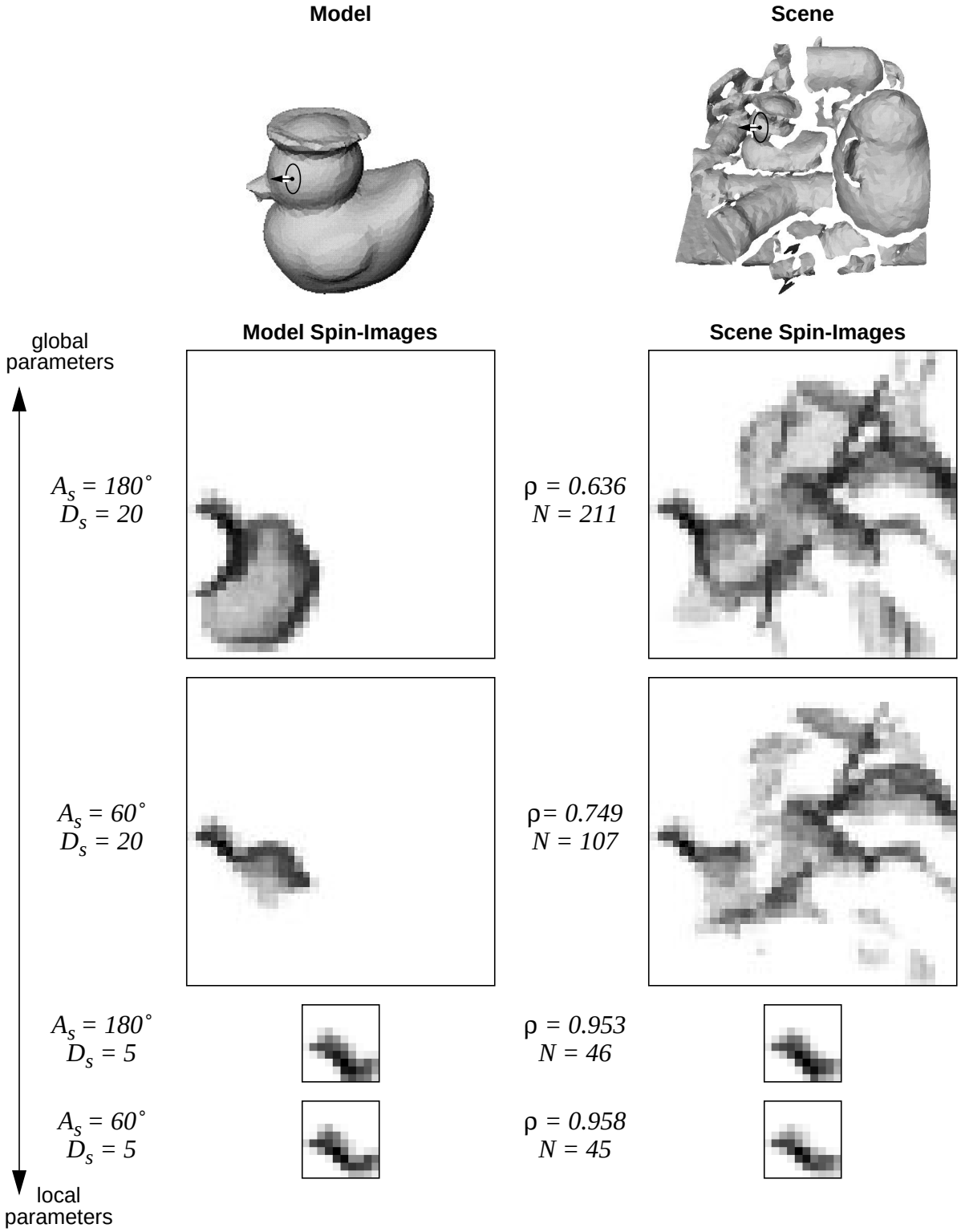


Figure 5-3: How spin-image generation parameters localize spin-images to reduce the effect of scene clutter on matching. On the right are shown the model and the model spin-images and on the left are shown the scene data and scene spin-images. As the spin-images become more localized, by decreasing A_S and D_S, the spin-images become more similar as indicated by the correlation coefficient ρ of the spin-

5.2.2 Handling Clutter

Often, the object that is being recognized is located in an image of a scene with clutter, scene data that is not from the model. Since all of the scene points are used to generate the scene spin-images, clutter can cause incorrect bin values due to points that are not on the model being spin-mapped into the spin-image. In other words, clutter in the scene is mapped into clutter in the spin-images. It is intuitive that the greater the distance between two points, the less likely that the two points are from the same object. This idea can be incorporated into the generation of spin-images to limit the effects of scene clutter on their appearance.

The simplest way to diminish the effects of scene clutter is to place a threshold on the maximum support distance D_s that is allowed between the oriented point basis of a spin-image and the points that are contributing to the image. If a scene point is not spin-mapped into the bounds of the image (as determined by image width) then the scene point does not contribute to the scene spin-image. Support distance places a cylindrical volume around the scene oriented point from which scene points are collected. Short support distances are employed to limit the effects of scene clutter. However, as the support distance decreases, the descriptiveness of the spin-images will decrease as well, because less of the overall shape of the object will be encoded. This problem is similar to that of reducing window size in correlation-based stereo. The effect of support distance on matching spin-images is described in detail in Chapter 9.

Figure 5-3 shows how spin-image generation parameters localize spin-images to reduce the effect of scene clutter on matching. When the spin-images are not localized, the model and scene spin-images are very different in appearance, a fact which is born out by the correlation coefficient of the two images. As the spin-images are localized by decreasing the support angle A_s and support distance D_s, the spin-images become much more similar.

Figure 5-4 explains this point in more detail. When creating spin-images with large support angle and distance, many scene points that do not belong to the model are spin-mapped into the scene spin-image. This causes the scene spin-image to become uncorrelated with the model spin-image because, as shown in the scatter plot of the two images, scene spin-image pixels are being corrupted by clutter. When smaller support angle and distance are used, the spin-images become similar; the pixel values shown in the scatter plot of the images created with local parameters are linearly related. By varying spin-image generation parameters, we are using

knowledge of the spin-image generation process to eliminate outlier pixels, making the spin-images much more similar.

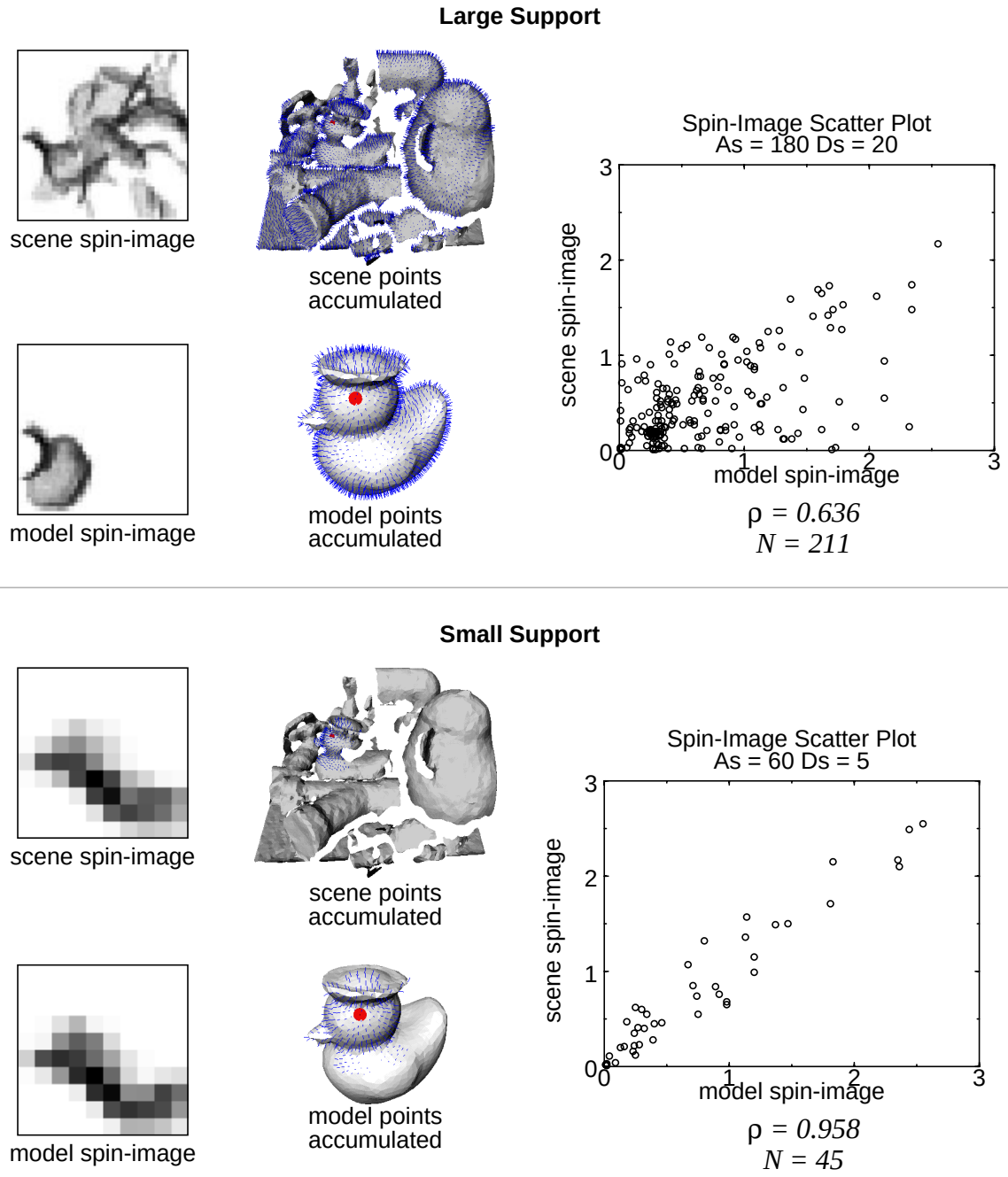


Figure 5-4: Localizing generation parameters increases the similarity of spin-images. The top shows a scatter plot of the model and scene spin-images generated using global parameters. The scatter plot shows that the spin-images are not particularly correlated. The bottom shows a scatter plot of the model and scene spin-images generated using local parameters. The scatter plot shows that the spin-images are much more correlated. Localizing the spin-images throws away image pixels where the images disagree.

5.3 Clutter Model

Any object recognition system designed for the real world must somehow deal with clutter and occlusion. Some systems perform segmentation before recognition in order to separate clutter from interesting object data. Our algorithm does not rely on segmentation. Instead, our system limits the effect of clutter and occlusion by localizing the support of spin-images used in matching. Spin-image support is controlled using two thresholds: support distance and support angle. Based on these thresholds, this section first develops a geometric model that shows that clutter is limited to connected regions in spin-image and therefore bounded. Using the boundedness of corruption in spin-images, we then show that the correlation coefficient is tolerant to limited amounts of clutter when used to compare spin-images.

5.3.1 Geometric Model

In order to geometrically analyze the effects of clutter, we have developed a simple model of the occurrence of clutter in spin-images by modeling objects as spheres. In this section, we present the important components of the model; we explain the details in Appendix C. Suppose

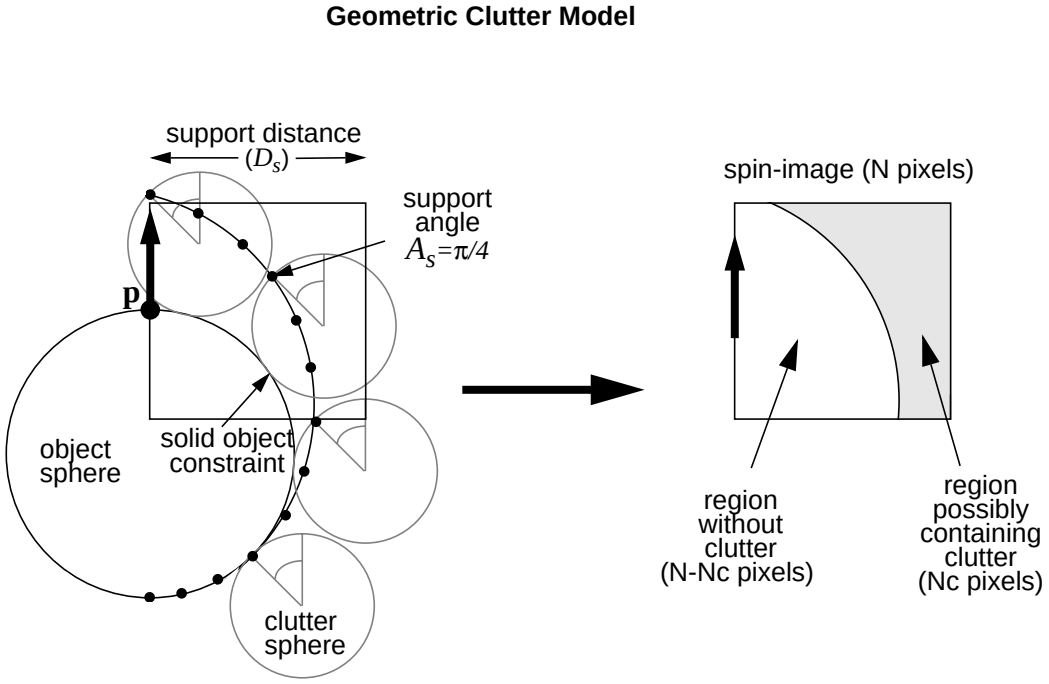


Figure 5-5: Theoretical clutter model. Because solid objects cannot intersect, and the support of spin-imag limited, the corruption of pixels due to clutter is limited to connected regions in spin-images

the scene contains an object to be recognized and an object that is creating clutter. Suppose we are generating a spin-image for an oriented point ***p*** (as shown in Figure 5-5) on the surface of an object in the scene that we would like to recognize. When modeling the effect of clutter on the spin-image, an appropriate sphere for the object is the largest sphere contained entirely within the object that also intersects ***p***. This sphere defines the largest volume of space around ***p*** that *must* be free of clutter. Similarly, suppose there exists a cluttering object in the scene. The appropriate size for the spherical model of the clutter object is determined as follows. Select a point on the clutter object. Find the largest sphere that touches the point and is still contained within the clutter object. Take the minimum diameter of these spheres over all points on the clutter object and you have the appropriate diameter for the spherical model of the clutter object. This sphere represents the closest *possible* contact between the object and the cluttering object. Since these spherical definitions place a lower bound on the space taken up by the object and an upper bound on the space taken up by the clutter, they are sufficient for determining the worst case effect of clutter on spin-images.

Clutter is limited to bounded and connected regions in spin-images.

Our model of the effect of clutter on spin-image generation is shown (in 2-D) in Figure 5-5. Suppose we would like to determine the spin-image for an oriented point **p** on an object modeled as a sphere as defined above. A point on the clutter object can be projected into the spin-

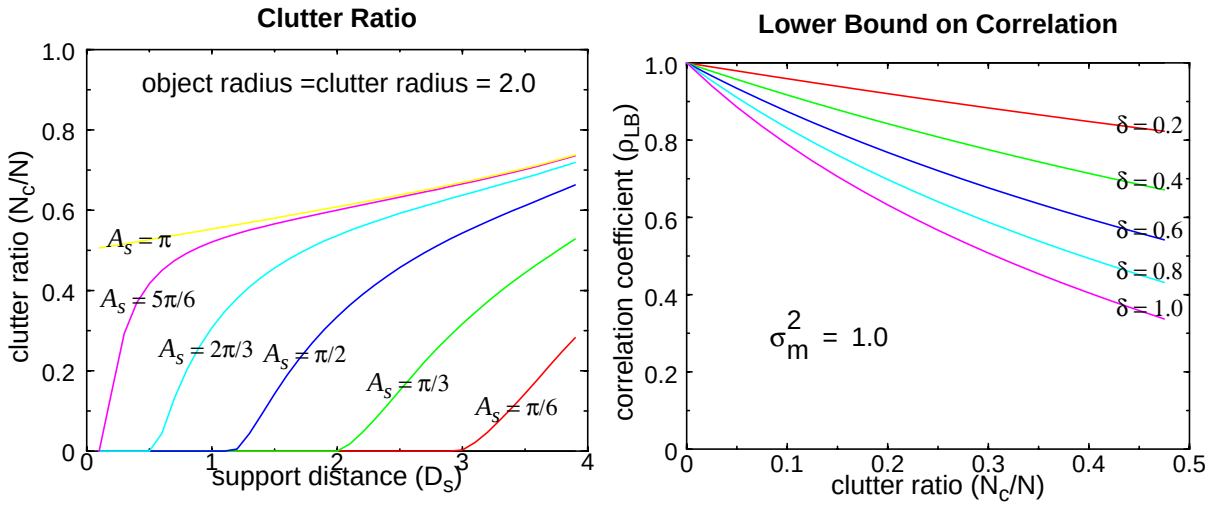


Figure 5-6: Clutter model plots. The effect of spin-image size and surface normal angle threshold on the percent of clutter in a scene spin-image is shown on the left and the worst case effect of clutter on correlation coefficient is shown on the right.

image of the oriented point if: the clutter point is within the support distance D_s, the clutter object does not intersect the object, and the angle between the surface normal of the clutter point and the oriented point is less than the support angle A_s. These three constraints describe a connected region in space that corresponds to a connected region in the spin-image, where clutter *might* possibly occur. It follows that clutter can affect only a limited number of the pixels in a spin image, and the effect of clutter can be limited by setting the thresholds stated above. The exact number of pixels affected by clutter (N_c) in a particular spin-image can be computed by checking which pixels in the spin-image satisfy the clutter constraints stated above. Let the clutter ratio be the percentage of pixels that can contain clutter N_c to the total number of pixels in the image N. Figure 5-6 contains a plot showing clutter ratio as a function of spin-image size D_s for different values of the angle threshold A_s. From the plot it can be seen that choosing low values for D_s and A_s will produce spin-images that contain no clutter. This analysis says nothing about the descriptiveness of spin-images that are constructed to be insensitive to clutter. Analysis of the effect of support angle and support distance on spin-image descriptiveness is given in Chapter 9.

5.3.2 Effect of Clutter on Correlation Coefficient

The above geometric model shows that the number of pixels in a spin-image that can be corrupted is bounded. Given the number of corrupted pixels in a spin-image is bounded, it is possible to predict a lower bound on the correlation coefficient, and therefore determine the effect of clutter and occlusion on the matching of spin-images. The following derivation is given in more detail in Appendix C. Suppose there exists a model surface mesh and a scene surface mesh that contains a copy of that model surface mesh. In this case, spin-images of corresponding points in the model and scene will be the same except for those pixels in the scene that are corrupted by clutter and occlusion. Let the subset of the scene spin-image (with N total pixels) that is corrupted by clutter be I_C (with N_C pixels), and the portion that is not corrupted be I_N. Let the scene spin-image have pixels x_i^s, the model spin-image have pixels x_i^m and the corruption at each scene pixels be δ_i. Then the relationship between model and scene pixels is

$$\text{if } x_i^s \in I_N \text{ then } x_i^s = x_i^m, \text{ and if } x_i^s \in I_C \text{ then } x_i^s = x_i^m + \delta_i \tag{5.1}$$

Clutter and occlusion manifest as extra and missing points in the scene where the number of

these points is bounded. Therefore, based on the spin-image generation process, the total change of any pixel in a scene spin-image that is corrupted by δ_i is bounded $|\delta_i| \le \delta$. Furthermore, the model and scene pixel values can be normalized on [0,1] with no effect on the correlation coefficient computed. Substitution of (5.1) into the definition of correlation coefficient

$$\rho = \left(\frac{1}{N}\sum x_i^m x_i^s - \frac{1}{N^2}\sum x_i^m \sum x_i^s\right) \Big/ \left(\left(\frac{1}{N}\sum (x_i^m)^2 - \left(\frac{1}{N}\sum x_i^m\right)^2\right)\left(\frac{1}{N}\sum (x_i^s)^2 - \left(\frac{1}{N}\sum x_i^s\right)^2\right)\right)^{\frac{1}{2}} \tag{5.2}$$

and using the normalization of pixel values and the bound on corruption, results in a lower bound on the correlation coefficient when comparing model and scene spin-images

$$\rho_{LB} = \left(\sigma_m^2 - \frac{N_C}{N}\delta\right) \Big/ \left(\sigma_m \sqrt{\sigma_m^2 + \frac{N_C}{N}(\delta + \delta^2)}\right) \tag{5.3}$$

In (5.3), σ_m^2 is the variance of the pixels in the model spin-image. Figure 5-6 shows plots of ρ_{LB} versus clutter ratio for increasing values of δ. From the plot it is clear that as clutter increases,

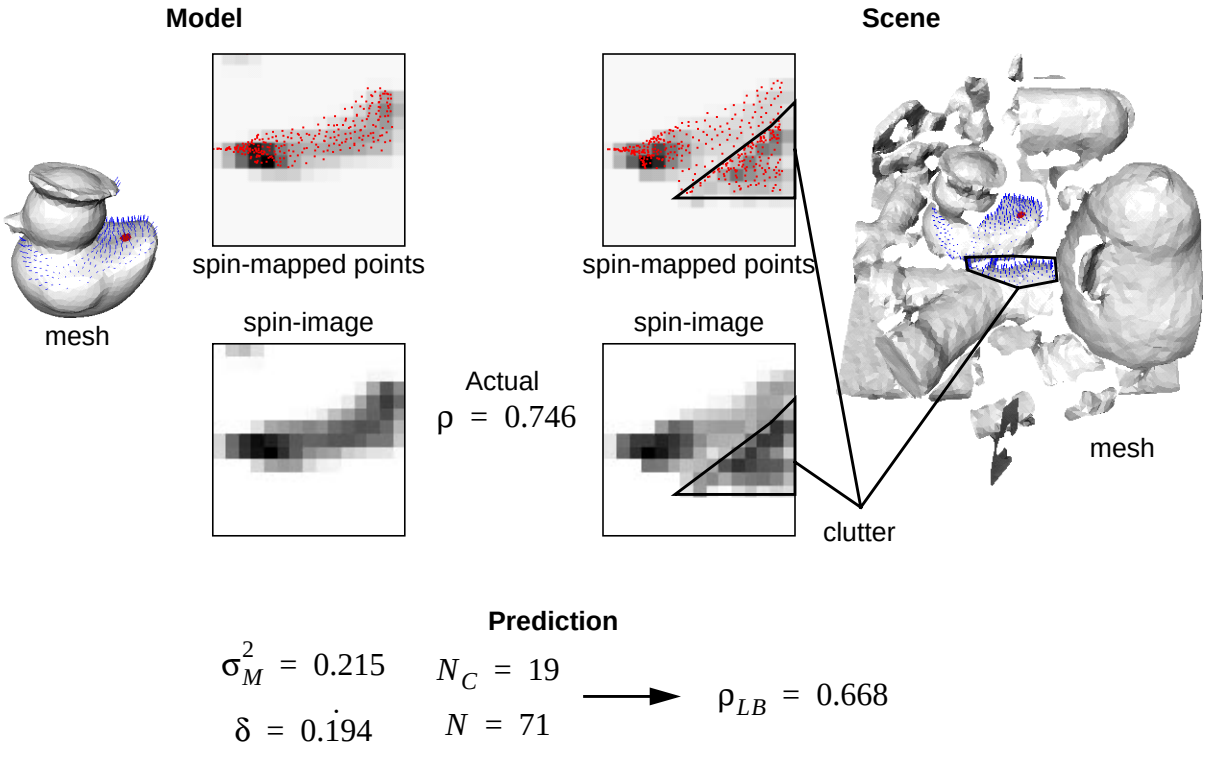


Figure 5-7: Experimental verification of lower bound on correlation coefficient in the presence of clutter. 19 of 71 pixels in the scene spin-image are corrupted by an amount δ less than 0.435. The correlation coefficient for the two spin-images of 0.746 is well above the lower bound of 0.301 predicted by the clutter model.

the worst case effect on correlation coefficient decreases gradually. This important result shows that our algorithm for matching spin-images does not fail catastrophically with the addition of clutter. Instead, the performance of our matching algorithm will gradually degrade as clutter and occlusion increase in the scene.

As clutter and occlusion increase in the scene, spin-image matching degrades smoothly.

Figure 5-7 validates the lower bound on the correlation coefficient in the presence of clutter and occlusion in a real scene. The correlation coefficient of the two corresponding images is 0.746, and the predicted lower bound on correlation coefficient computed using (5.3) is 0.301 which is well below the correlation coefficient obtained. This result demonstrates that our model lower bound on correlation coefficient is very conservative. Even though the bound is conservative, Figure 5-6 still shows that a high correlation coefficient can be obtained with moderate amounts of clutter.

Chapter 6

Spin-Image Compression

Spin-images are a highly redundant representation of surface shape. The next chapter explains how this redundancy can be exploited to compress the representation of spin-images to reduce the storage and speed up the matching of spin-images. As will be shown, spin-image compression is based on principal component analysis, a common technique in image compression and object recognition.

6.1 Redundancy of Spin-Images

Upon inspection, one notices that there is a great deal of redundancy between spin-images coming from the same surface. Most of this redundancy stems from the fact that spin-images generated from two oriented point bases that are close to each other on the surface will be highly correlated. If two oriented point bases are close to each other, then the spin-map coordinates of the other vertices on the model with respect to the two different bases will be similar, causing

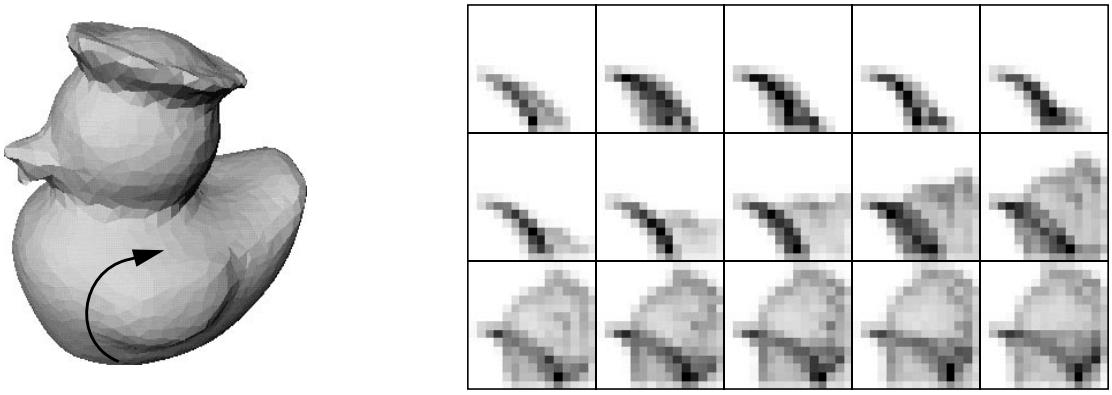


Figure 6-1: Spin-images generated while traversing a path along the surface of the duckie model. Spin-images from proximal oriented points are similar, resulting in one cause of redundancy in spin-images.

the spin-images generated to be similar. In fact, if the support of two spin-images contain exactly the same surface points, then the only variation between spin-images will be the differences in the oriented point bases defining the images. Since oriented points have only five parameters and spin-images are usually on order of 100 pixels, a great deal of redundancy will obviously exist. Figure 6-1 demonstrates the redundancy of spin-images on a model of a duckie. The sequence of spin-images shown was taken for vertices along a continuous path over the surface of the duck. In the first part of the path the position of the oriented point basis is changing while its surface normal is remaining relatively fixed. This results in small changes in the spin-images. In the second part of the path, both the position and orientation of the oriented point are changing, causing much greater changes in the spin-image shape as the path is traversed. Similarity between spin-images is more sensitive to surface normal position than it is to position of the vertex making up the oriented point.

Another possible cause of redundancy is symmetry in the surface. Surface symmetry is compounded by the cylindrical symmetry of the spin-image generation process. Two oriented point bases on equal but opposite sides of a plane of symmetry will be the same. This is demonstrated in Figure 6-2 where spin-images from bases on opposite sides of a plane of symmetry for the duckie model are shown. The duckie model has two planes of symmetry, so groups of four oriented points bases produce similar spin-images.

When discussing the redundancy of spin-images it is convenient to think of images as vectors living in an N-dimensional vector space when N is the number of pixels in the image. Each pixel, in raster order, determines an axis of the vector space, and a pixel value is the coordinate of the vector along the axis given by the raster position of the pixel. Let us call the vector space

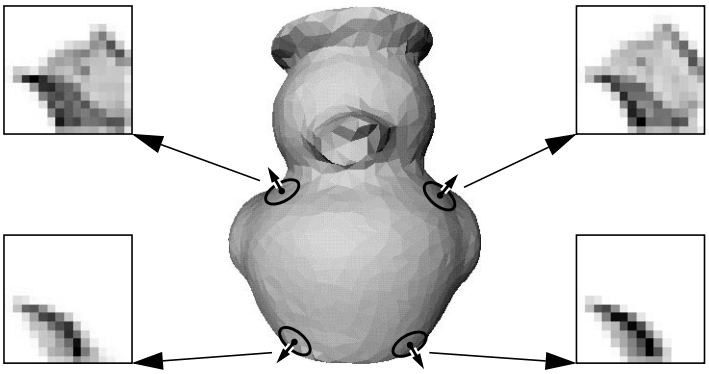


Figure 6-2: Two sets of four similar spin-images caused by symmetry in the duckie model.

containing the spin-images the s*pin-image-space*.

If the surface being described by a set of spin-images is not extremely complicated, then the spin-images produced will be correlated. This correlation can prevent the spin-images from spanning the spin-image-space, forcing the spin-images to exist in a lower dimensional subspace of the spin-image-space. Describing spin-images with respect to this lower dimensional subspace, if it exists, is the key to spin-image compression.

When spin-images are highly correlated, their distance in the spin-image-space will be small. Proximity of oriented points is a good indicator of correlation between spin-images. Since the connectivity of the surface mesh conveys the proximity of oriented points, it can also be used to convey the proximity of spin-images in the spin-image-space. In fact, just as surface mesh connectivity creates a piece-wise linear 2-D manifold in Euclidean space, this surface mesh connectivity can also be used to describe a continuous piece-wise linear 2-D manifold in which all spin-images for the surface live. This assumes that the spin-images for oriented points on faces between vertices on the mesh can be linearly interpolated from the spin-images generated at the vertices of the face. Although this manifold will have a contorted shape, based on the complexity of the model it may not span the entire spin-image-space. In the next sections, we will investigate the use of principal component analysis to concisely describe the space in which spin-images exist. We will then show how a technique similar to parametric appearance-based recognition [65] or “Eigen-faces” [94] can be used to make object recognition efficient and robust.

6.2 Model Compression

Since spin-images from a single model are highly redundant, they can be compressed by representing each spin-image by a small number of coefficients. This reduction of the spin-image to a small number of coefficients will result in fewer model storage requirements and faster and more robust spin-image matching.

6.2.1 Principal component analysis

A common technique for vector compression when vectors are highly redundant is principal component analysis (PCA). PCA or Karhunen-Loeve expansion [32] is a well known method

for computing the principal directions of a set of vectors. By computing the eigenvectors of the covariance matrix of the set of vectors, PCA determines an orthogonal basis, called the eigenspace, in which to describe the vectors. The eigen-space has some nice properties: Since the eigenspace is a orthogonal basis spanning the vector space, it preserves the structure of the set of vectors (i.e., distances and angles are preserved). Since the eigenvectors determine an orthogonal basis for the vector space, a vector can be transformed into the eigenspace by creating a vector from its projections onto the eigenvectors of the eigenspace.

The second useful property of the eigenspace is that the effectiveness of each eigenvector in describing the spread of a set of vectors is determined by its eigenvalue; eigenvectors of large eigenvalue will be directions along which the set of vectors varies the most. This property is fundamental to using PCA for vector compression. In vector compression, the idea is to describe a vector by as few coordinates as possible, thus reducing storage size of the vector, while minimizing the deviation (mean square error) from the true vector value. Since the eigenvectors are ordered in effectiveness by eigenvalue, representing a vector in the eigenspace by its projection onto the eigenvectors of large eigenvalue and truncating the projections onto the eigenvectors of low eigenvalue will result in the least deviation between the truncated vector and its true value. Over all choices of orthonormal bases, the eigenspace is the best basis for minimizing this deviation [32].

Principal Component Analysis has become popular in computer vision for comparison and compression of images because it is optimal in the correlation sense. What this means is that the l_2 distance between two images in their original coordinate system is the same as the l_2 distance between the two images represented in the eigenspace. Furthermore, when vectors are represented in the eigenspace, the l_2 distance between truncated vectors is the best approximation to the l_2 distance between the untruncated vectors with respect to mean square error. Thus, PCA gives us an elegant way to balance compression of images against ability to discriminate between images. Given a set of images to compress, PCA is used to compute the eigenspace of the set of images. The set of images is then stored as the s most significant eigenvectors of the eigen space and an s-tuple for each image representing its projection onto the s most significant eigenvectors. Similarity between images is determined by computing the l_2 distance between s-tuples representing the images. Thus, the amount of storage for the images and the time to

compare them is reduced. The dimension of the truncated eigenspace depends on the set of images and the required fidelity of reconstruction.

6.2.2 Model Eigen-Spin-Images

When used for feature classification, principal component analysis uses the assumption that the data is scattered in a hyperellipsoid shape. When this assumption is not justified, feature classification with PCA will not work as well. As mentioned in the previous section, spin-images

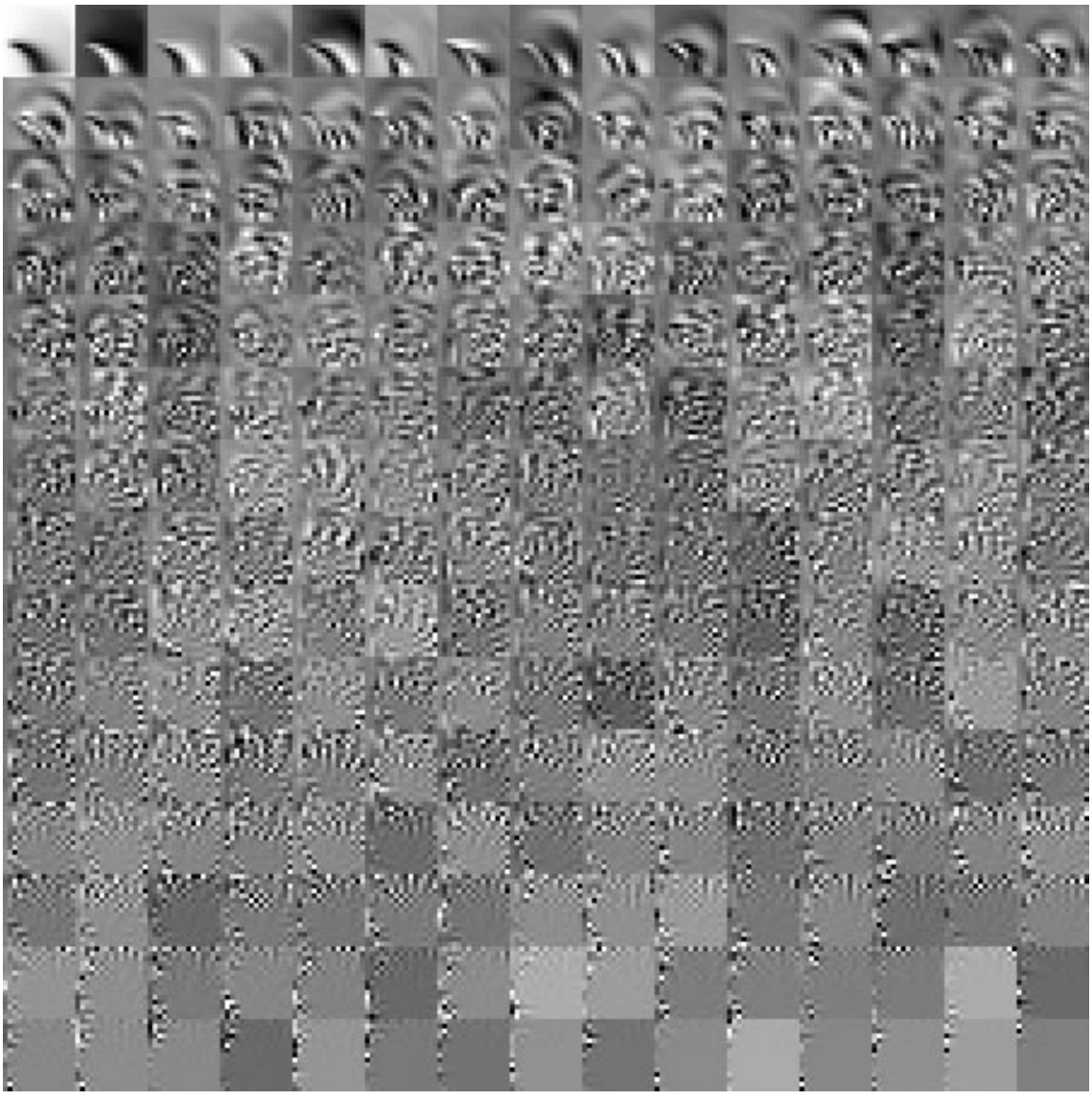


Figure 6-3: Eigen-spin-images (created using the parameters of the spin-images shown in Figure 6-1) in eigenvalue order for the duckie model. More significant spin-images convey the low frequency components in the spin-images while high frequency components are conveyed in the less significant eigen-spin-images. The spin-image in the upper left is the mean spin-image.

are very structured, so in the case of spin-images, the assumption that the data is hyperellipsoidal is certainly not justified. However, PCA is still useful because it determines which directions are important for describing the data and which directions are unimportant. If we use only this information and none of the information derived from the hyperellipsoid assumption (i.e., relative eigenvalues), then PCA will still be useful for spin-image compression.

Compressing the spin-images from a single model stack is the obvious first place that principal component analysis can be applied because the spin-images will contain points from the same model and will therefore be highly redundant. Suppose a model M has N spin-images $\boldsymbol{x_i}$ of size D, then the mean of all of the spin-images is

$$\bar{\boldsymbol{x}} = \sum_{i=1}^{N} \boldsymbol{x_i}. \tag{6.1}$$

Subtracting the mean of the spin-images from each spin-image makes the principal directions computed by PCA more effective for describing the variance between spin-images.

$$\hat{\boldsymbol{x}_i} = \boldsymbol{x_i} - \bar{\boldsymbol{x}} \tag{6.2}$$

The mean-subtracted stack of spin-images from the model can be represented as an DxN matrix with each column of the matrix being a mean-subtracted spin-image

$$S^m = \left[\hat{\boldsymbol{x}}_1 \ \hat{\boldsymbol{x}}_2 \ \ldots \ \hat{\boldsymbol{x}}_N\right]. \tag{6.3}$$

The covariance of the spin-images in the model is the DxD matrix C given by

$$C^m = S^m (S^m)^T. \tag{6.4}$$

The eigenvectors of C are then computed by solving the eigenvector problem

$$\lambda_i^m \boldsymbol{e_i}^m = C^m \boldsymbol{e_i}^m. \tag{6.5}$$

Since the dimension of the spin-images is not too large (~100), the standard **`jacobi`** algorithm from Numerical Recipes in C [71] is used to determine the eigenvectors $\boldsymbol{e_j^m}$ and eigenvalues $\lambda_{\boldsymbol{j}}^{\boldsymbol{m}}$ of C^m. Since the eigenvectors of C^m can be considered as spin-images, they will be called

eigen-spin-images.

Figure 6-3 shows the 100 eigen-spin-images of a model of a duckie in order of eigenvalue. Eigen-spin-images are even more difficult to visualize than regular spin-images, but some general properties can be gleaned from their appearance. First, the most significant eigen spin-images are noticeably smoother than typical spin-images. This is because the eigen-spin-images

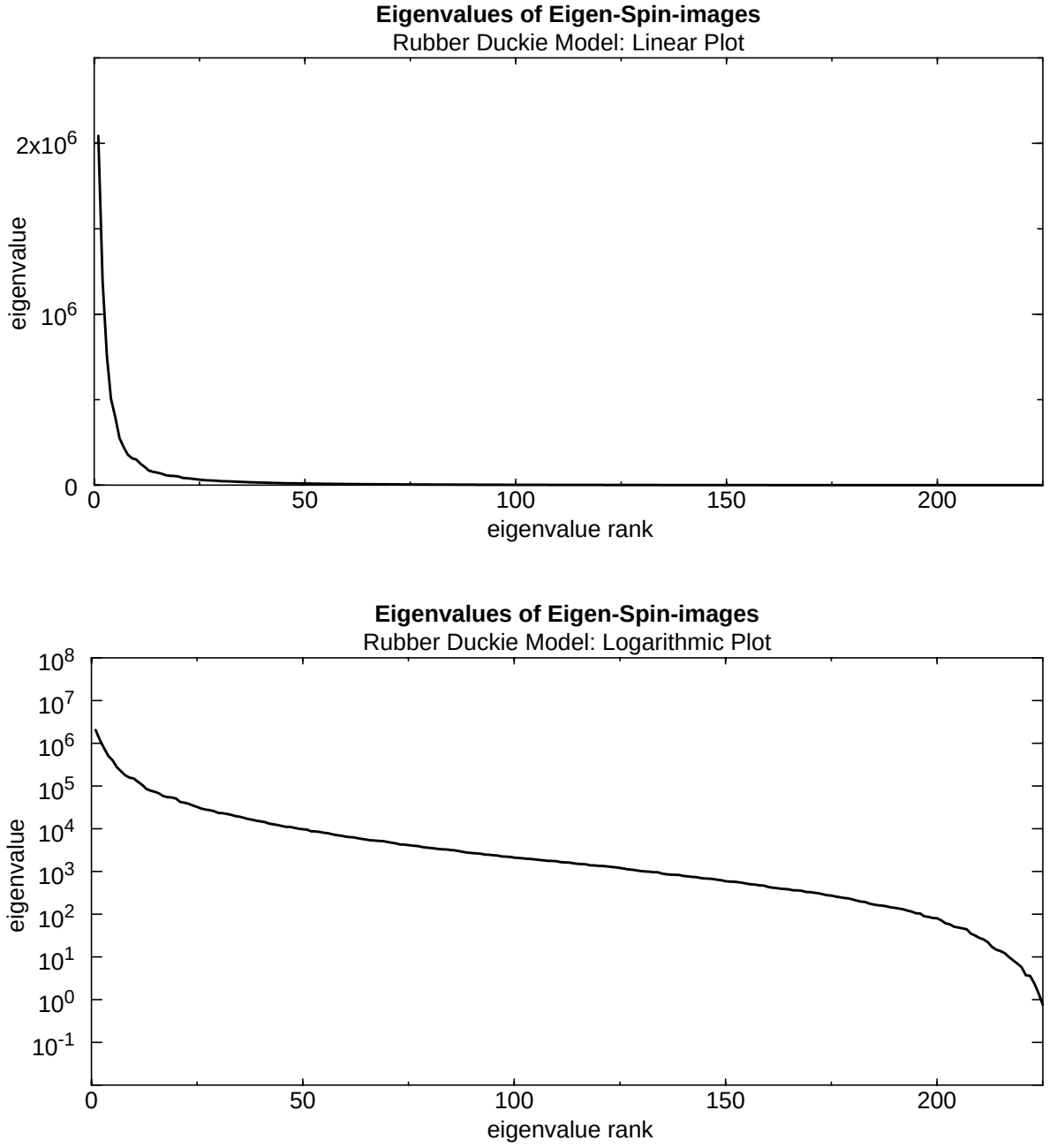


Figure 6-4: Linear and logarithmic plots of eigenvalues for the eigenvalues of the duckie model shown in Figure 6-3.

convey the global shape of all of the spin-images from the model. Second, some of the significant eigen-spin-images have the appearance of edge operators or other feature matching kernels. They are representing the low frequency components of the shape of the models that are important to the spin-images as a group. The lower order eigen-spin-images become much less smooth and variable than the high order eigen-spin-images which are representing the high frequency components in the spin-images. Even though the shape of the model may not have high frequencies, the spin-images will have high frequency components due to the spin-image generation process; vertices at distinct positions rather than the entire surface are used to create each spin-image. By truncating these eigen-spin-images, the effect of point distribution and noise can be reduced in the spin-image matching process.

Figure 6-4 shows a linear and logarithmic plot of the eigenvalues for the eigen-spin-images shown in Figure 6-3. After the first 20, the eigenvalues become negligible. This indicates that the model spin-images live in a small dimensional subspace of the eigenspace and can be accurately conveyed by projections onto a small number of eigen-spin-images. The exact method for determining where to cut off the projection dimension is discussed in Section 6.2.3.

Once the projection dimension s has been determined then the compressed model representation can be made. Each spin-image is replaced by an s-tuple of eigen-spin-image projections (6.7). For the tuples to be used in recognition, the first s eigen-spin-images and the mean spin image (6.1) must also be stored with the model. If the dimension of the projection s-tuple is much less than the dimension of the spin-image, then replacing spin-images with tuples will greatly reduce the amount of storage needed for each model. As will be explained in Section 6.3, reducing the dimension of the spin-image representation allows efficient techniques for matching spin-images to be introduced for increasing the spin-image matching speed.

6.2.3 Determining Model Projection Dimension

Eigenvalues cannot be used to compute the cutoff for the dimension of the projection because this would make implicit the assumption that the spin-images are distributed in a hyperellipsoid. Since the distribution of the spin-images is undoubtedly more complicated, a more sophisticated technique based on spin-image reconstruction is needed to determine the cutoff.

First, some notation needs to be developed. Given a mean-subtracted spin-image $\hat{\boldsymbol{x}}_i$, let its projection onto the *j*th eigen-spin-image $\boldsymbol{e}_j^m$ be

$$p_{ij} = \hat{\boldsymbol{x}}_i \cdot \boldsymbol{e}_j^m \tag{6.6}$$

An *s*-tuple of projection coordinates can then be created for each spin-image

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{im}). \tag{6.7}$$

The order *s* reconstruction of $\hat{\boldsymbol{x}}_i$ is then

$$\hat{\boldsymbol{x}}_i^s = \sum_{j=1}^{s} \boldsymbol{e}_j^m p_{ij} \tag{6.8}$$

Determining the projection cutoff requires finding a small value for *s* that will result in sufficient reconstruction of spin-images. Correlation coefficient should be used to determine the goodness of the reconstruction because of its normalization properties that make the comparison independent of image mean and variance. The matching engine also compares spin-images based on correlation coefficient, so any cutoff determined using correlation coefficient will be consistent with the matching engine. Formally, the reconstruction error function ρ_{RE}^m is the

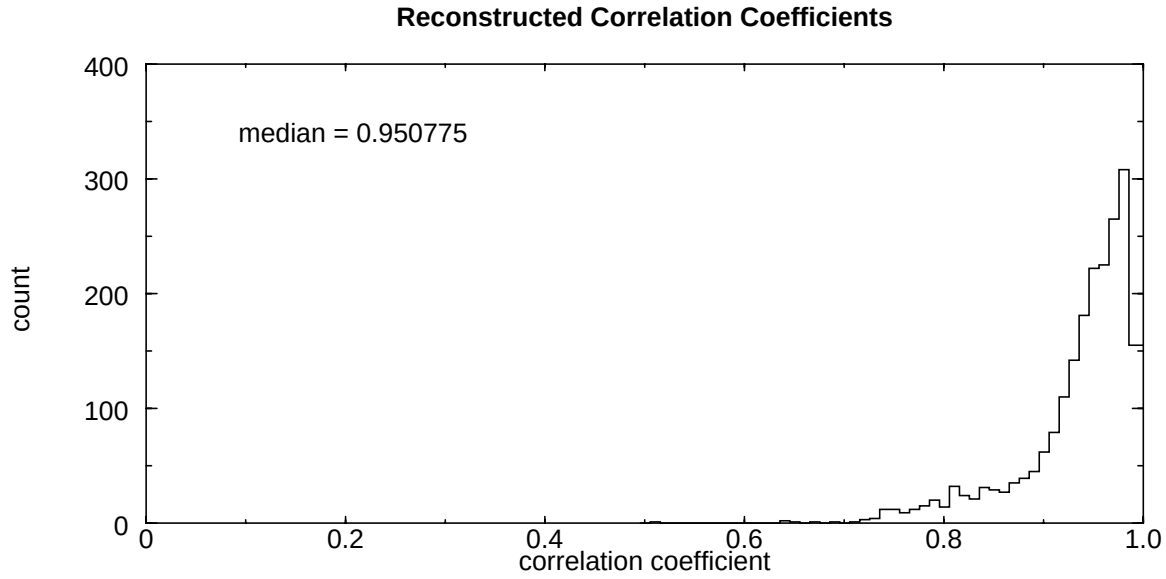


Figure 6-5: Reconstruction correlation coefficients for the duckie model at a level of 0.95 which resulted in a projection dimension of 11, an order of magnitude compression (11/225 = 5%) in the spin-image representation.

correlation coefficient between the original mean-subtracted spin-image and its order *s* reconstruction

$$\rho_{RE}^{s} = \rho(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_i^{s}). \tag{6.9}$$

Ideally, the cutoff would be set such that ρ_{RE}^{s} is greater than some predetermined value for all of the spin-images in the model. In practice, this is not feasible because it would require projection onto a large number of eigen-spin-images, thus removing the benefits of compression. Instead, the best projection cutoff m_v is determined as the s_v for which the median of ρ_{RE}^{s} is greater than a predetermined value *v* between -1 and 1.

$$(m_v \,|\, (\underset{i}{Median}\ (\rho_{RE}^{s_v}(\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{x}}_i^{s_v})) > v)) \tag{6.10}$$

Using the median operation will guarantee that at least half of the spin-images will be reconstructed such that their correlation coefficient with the original spin-image will be greater than the value specified. The median measure will also prevent outliers in the reconstruction, caused by unusual spin-images, from influencing the cutoff dimension. Since *v* is a measure of correlation coefficient, it is typically set between 0.90 and 0.99 to guarantee good reconstructions. Figure 6-5 shows the reconstructed correlation coefficients for the model of the duckie at a reconstruction level of 0.90. In this case (~20 of 200) eigen-spin-images were used in the reconstruction. For this model an order of magnitude compression in the spin-image representation is possible at a reconstruction level of 0.90.

Figure 6-6 shows the reconstruction of spin-images for the duckie model as the number of eigen-spin-images is increased. Reconstruction correlation coefficient is color coded on the

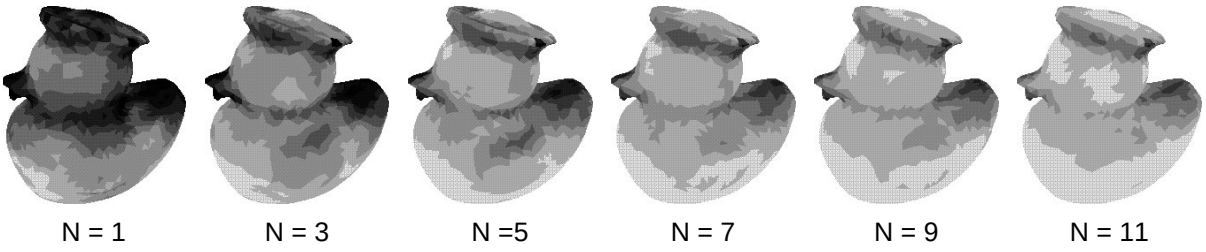


Figure 6-6: Reconstruction correlation coefficient plotted on the duckie model for increasing number of eigenvectors.

models: white corresponds to 1.0 and black corresponds to 0.0. This display gives insight into the contribution of the eigen-spin-images to the reconstruction of the spin-images. The low order eigen-spin-images are good at reconstructing the gross shape of the model and bad at reconstructing the parts of the model with more detail. As the number of eigen-spin-images in the reconstruction increases, the reconstruction of the detailed areas improves. The last reconstruction shows the duckie reconstructed with eleven eigen-spin-images. The area around the top of the duckie is darker than the rest of the model. The spin-images constructed from oriented points in this part of the duckie will not be reconstructed as well as the spin-images in other parts of the duckie because of unusualness of the spin-images in this area. An interesting extension of this work will be to somehow invert the reconstructed spin-images to create a prototypical shape model of the reconstructed object.

6.3 Matching Spin-Images with Compression

This section explains how scene spin-images are matched to compressed model spin-images represented as *s*-tuples. Given the low dimension of *s*-tuples, it is possible to match spin-images in time that is sub-linear in the number of spin-images in the model using efficient closest point search structures.

6.3.1 The Importance of Mesh Resolution to Tuple Matching

Unlike when matching spin-images without compression, mesh resolution (defined in Section 2.1.1) is very important when matching spin-images with compression. The vector magnitude of a spin-image depends on the resolution of the surface mesh it was created from because resolution controls the number of points falling into each bin of the spin-image. If the resolution of the mesh is higher, then more points will fall into each bin, resulting in a greater vector magnitude for the spin-image, even after the spin-image is normalized by its variance. When spin-images from two meshes of the same surface shape, but of different resolution, are being compared, the spin-image from the higher resolution mesh will have a greater magnitude than the spin-image from the mesh with lower resolution. However, the vector direction of the spin-images will be the same. This difference in magnitude caused by difference in mesh resolution will not hinder matching of uncompressed spin-images because normalized correlation

coefficient is used (normalized correlation coefficient subtracts out the means and divides by the standard deviation of the spin-images during comparison, eliminating the difference in magnitude between the spin-images). During compression, any difference in magnitude between spin-images will be directly translated to differences in magnitude between *s*-tuples constructed from the spin-images. Unfortunately, differences in *s*-tuple magnitude will directly affect the matching of *s*-tuples because the l_2 distance, which does no normalization, is used in the comparison.

To compensate for the dependence of *s*-tuple matching on spin-image magnitude, the scene surface mesh and all of the models in the model library should be of similar resolution. This will prevent differences in magnitude between the model *s*-tuples and the scene *s*-tuples and will guarantee better matching. Since there exist solutions to controlling the resolution of meshes guaranteeing similarity in mesh resolutions will not be a problem. Future work will look into efficient storage of s-tuples that are also independent of the magnitude of the spin-

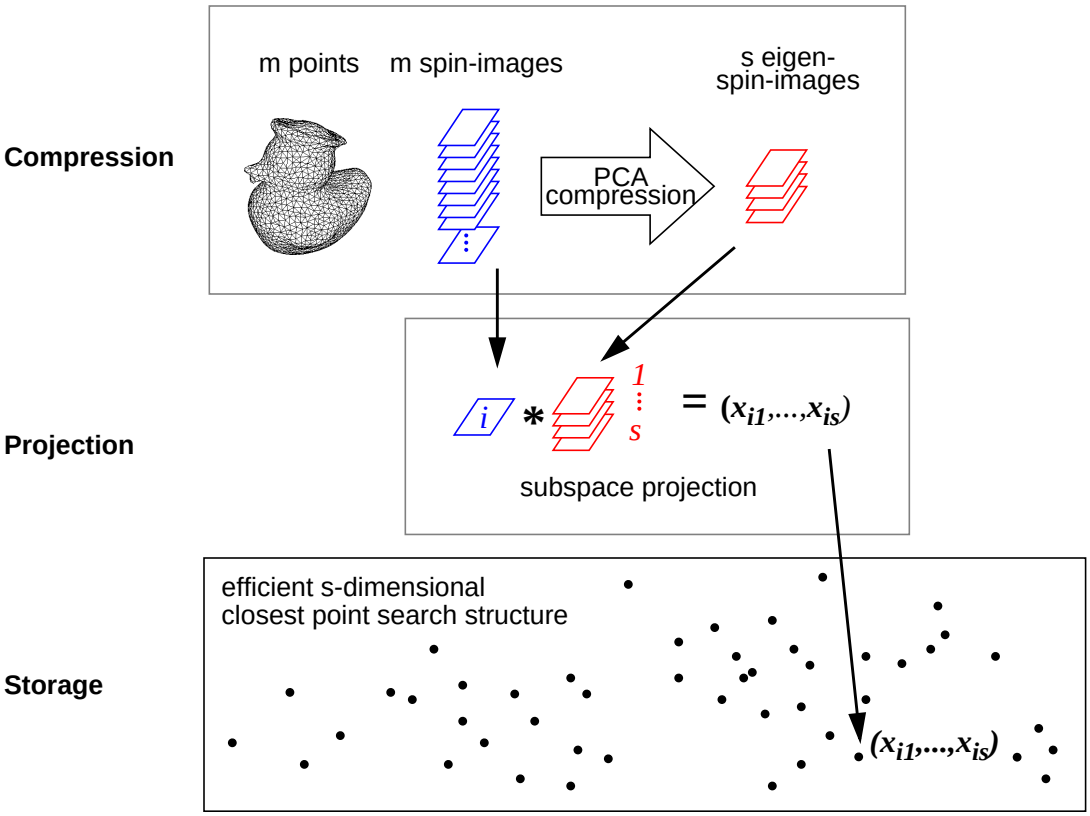


Figure 6-7: Compression of a stack of model spin-images. PCA is used to determine the most important directions for discriminating between spin-images. Spin-images are replaced by their projection onto these most important directions.

images being compared.

6.3.2 Constructing Scene Tuples

To match a scene spin-image to a model *s*-tuple, a scene *s*-tuple must be generated for the scene spin-image. Suppose the scene spin-image for scene oriented point *j* is represented in vector notation as $\boldsymbol{y}_j$. The first step in constructing the scene *s*-tuple is to subtract the mean of the model spin-images

$$\hat{\boldsymbol{y}}_j = \boldsymbol{y}_j - \bar{\boldsymbol{x}} \tag{6.11}$$

Next the mean-subtracted scene spin-image is projected onto the top s model eigen-spin-images to get the scene *s*-tuple $\boldsymbol{t}_j$

$$\boldsymbol{t}_j = (\hat{\boldsymbol{y}}_j \boldsymbol{e}_1^{\boldsymbol{m}}, \hat{\boldsymbol{y}}_j \boldsymbol{e}_2^{\boldsymbol{m}}, \ldots, \hat{\boldsymbol{y}}_j \boldsymbol{e}_s^{\boldsymbol{m}}) \tag{6.12}$$

The scene *s*-tuple is the projection of the scene spin-image onto the principal directions of the model spin-images.

6.3.3 Finding Closest Model Tuples

To determine the best matching model spin-image to scene spin-image, the l_2 distance between the scene and model tuples is used. As discussed above, the model eigenspace preserves distances, so the distance between spin-images in the spin-image space will be the same as the distance between spin-images in the eigenspace. When the eigenspace is truncated, the distance between points will still be a good approximation of the distance between spin-images in the spin-image space. Although the l_2 distance between spin-images is not the same as the similarity measure used in spin-image matching, it is still a good measure of the similarity of two spin-images. The l_2 norm neglects the second part of the similarity measure which uses the overlap between spin-images as a measure of similarity. This will cause the l_2 norm to be a less accurate measure of similarity. However, the matching efficiency gained by using the l_2 norm, through the use of efficient closest point search structures, out weighs this loss of accuracy.

The least efficient way to compare *s*-tuples is to exhaustively compare the scene *s*-tuple to all of the model *s*-tuples. This method is guaranteed to find the closest model *s*-tuple to the scene

s-tuple. Given the reduction in dimension of the tuples over spin-images, the search time for the closest matching model point will be much less when using *s*-tuples than when using spin-images. However, since *s* is usually small (*s~20*), there exist more efficient ways to organize the *s*-tuples for efficient closest point search.

One possible method is to store the model *s*-tuples in a *s*-dimensional kD-tree [5][31]. KD-trees organize vectors for efficient closest point search by partitioning the space of vectors with dynamically determined cutting planes. Unfortunately, the look up efficiency $O(DN^{1-1/D})$ [70] of kD-trees decreases with the dimension *D* of the vectors. At the dimension of typical *s*-tuples (20 to 40), kD-trees are not significantly more efficient than exhaustive search. Furthermore, kD-trees work best when the data is unstructured, which is certainly not the case with *s*-tuples. After experimenting with kD-trees for efficient closest *s*-tuple search, it was confirmed that kD-trees did not provide search times that were significantly better than exhaustive search, so they were discounted as structures for efficient closest *s*-tuple search.

Another structure for efficient closest point search was recently proposed by Nene and Nayar [67]. The efficiency of their data structure is based on the assumption that one is only interested in the closest point, if it is less than a predetermined distance ε from the query point. With this assumption, they have come up with a data structure for closest point search whose efficiency, is independent of the dimension of the points. If ε is small, the search complexity is *O(N)* with *N* points coming from uniform or normally distributed distributions.

The algorithm works as follows. First, the points are sorted on each coordinate. For each sort, a forward and backward map that keeps track of the sorted and unsorted positions of the points is created. At search time, the user needs to input a maximum search distance ε beyond which no closest point will be returned. If the points have dimension *s*, then closest point search has s stages, one for each dimension. Initially all points are put on the keep list. During stage *i*, the points in the keep list whose *ith* coordinates are within ε of the *ith* coordinate of the query point are retained on the keep list; the others are removed using the forward and backward maps. After cycling through all of the dimensions, the points left on the keep list are the ones that are within ε, in the l_1 norm, of the query point. In the original algorithm, the closest point in the l_2 norm, if one exists, is returned as the closest point. For s-tuple matching, it was found that returning all of the points that are within ε, in the l_1 norm, of the query *s*-tuple produced more

robust recognition results for the same reasons that spin-image matching returns multiple closest spin-images.

Order of magnitude improvements in the time to search for closest points were achieved by using this algorithm over exhaustive search (see Figure 7-30), so it was adopted as the closest point search structure used for matching *s*-tuples. The applicability of the algorithm to the problem of matching *s*-tuples is not surprising; the authors of the algorithm demonstrated its effectiveness in the domain of appearance based recognition [65], a domain that is similar to spin-image matching. In both domains, PCA is used to compress images resulting in set of structured *s*-tuples that must be searched for closest points.

The parameter ε determines the threshold on the closest *s*-tuples returned, so it is aN important parameter. Furthermore, the efficiency of the closest point search depends on setting ε as small as possible. Fortunately, an appropriate range of values for ε can be determined by analyzing the distance between model *s*-tuples. Since the scene has the same resolution as the models, it is feasible that the difference between scene spin-images and matching model spin-images will be on order of the distance between a model spin-image and the other model spin-image that is closest to it. Using optimality arguments given in Section 6.2.1 it follows that the distance between closest model *s*-tuples will be on order of the distance between scene *s*-tuples and model s-tuples. Therefore, by measuring the distance between closest model *s*-tuples, a scale for setting ε is obtained. Theoretically, using the median of the distance between all pairs of closest model *s*-tuples will produce a robust estimate for ε which will be large enough to acquire the closest *s*-tuple, but still small enough that the algorithm will be efficient. In practice it was found that setting ε to one to two times the median of the distance between all pairs of closest model *s*-tuples resulted in more matches being produced and better recognition results.

In *s*-tuple matching, ε plays a similar role to the dynamic outlier threshold used in spin-image matching. However, ε is not set dynamically because this would require exhaustive computation of the distance between the scene and all model *s*-tuples, a computation avoided in an the efficient closest point search algorithm. Herein lies a trade-off, efficient search versus robust detection of matches. The recognition results shown in the next chapter convey that the possible loss in recognition accuracy is not significant given the large increase in recognition speed.

6.3.4 Algorithm for Spin-Image Matching with Compression

Using spin-image compression improves the running time of the recognition algorithm by speeding up the search for model points that match scene points. Otherwise, the recognition algorithm with compression is very similar to the recognition algorithm without compression. Before recognition, the spin-images for each model in the model library are computed. Then, the eigen-spin-images for each model are computed. Next, the projection dimension is determined separately for each model. Next, the *s*-tuples for the spin-images in each model are computed by projecting model spin-images onto model eigen-spin-images. Finally, model *s*-tuples are then stored in an efficient closest point search structure. The model preprocessing is shown pictorially in Figure 6-7.

At recognition time, a percentage of oriented points are selected at random from the scene.

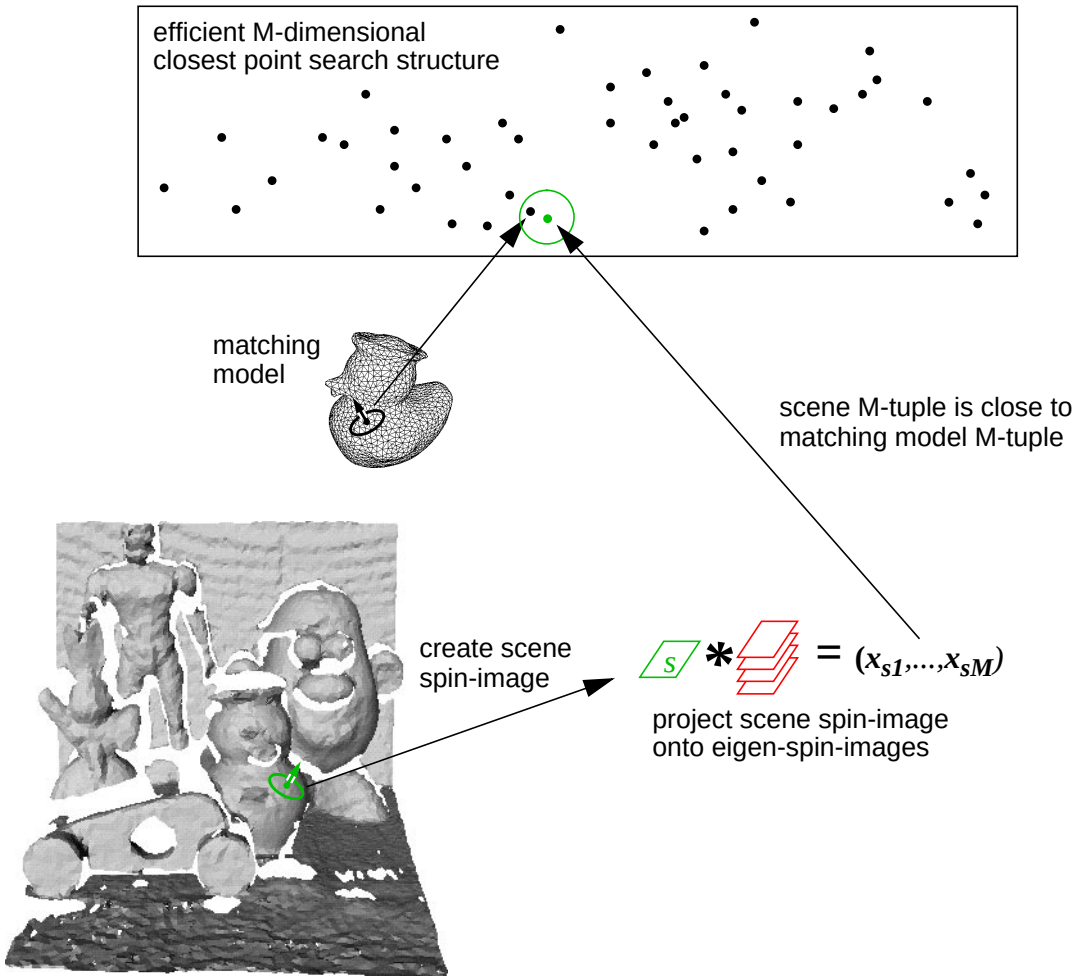


Figure 6-8: Matching oriented points with compressed spin-images.

Next each selected scene point is compared sequentially to each model in the model library as follows. First, the scene point's spin-image is created using the current model's spin-image parameters and the scene data. Next, the scene spin-image is projected onto the current model's eigen-spin-images to obtain a scene *s*-tuple. The scene *s*-tuple is then used as a query point into the current model's efficient closest point search structure which returns a list of current model *s*-tuples close to the scene *s*-tuple. These matching s-tuples are then used to create possible oriented point correspondences between the scene and the current model. This procedure is repeated for all models resulting in oriented point correspondences between the scene and possibly multiple models which are fed into the surface matching engine. A pictorial description of the on-line recognition using compressed spin-images is given in Figure 6-8. Although the decrease in recognition times produced by spin-image compression are significant, the complexity of the algorithm is still linear in the number of models. In the next section a more efficient method for determining the matching model before determining the matching model vertex is described.

6.4 Library Compression

The running time of the recognition algorithm described in the previous section grows linearly with the number of models in the model library; each additional model requires an additional search for closest matching model spin-images. For large model libraries, this is an unacceptable growth rate. This section describes a method for quickly determining the correct model matching a scene spin-image based on spin-image compression using principal component analysis.

In Section 6.2.2, it was shown how principal component analysis can be used to compress the spin-images coming from a single model. In that case, model eigen-spin-images were used to determine the principal directions of variation between spin-images from the same model, so that points on the model could be differentiated in the best way possible. In other words, model compression is used for localizing the best matching point on a model. PCA can also be used to compress spin-images from many models in order to quickly determine the best model matching a scene spin-image. This form of compression is called *library compression*. In library compression, the goal is different: to compress spin-images in such a way that increases

the differentiation between models but not necessarily between points on the model. Thus, library compression is used for recognition more than for localization.

Following the ideas of the "universal-eigenspace" developed by Murase and Nayar [65], principal component analysis can be used find the best compression of spin-images for differentiation between models. The general idea is to determine the eigenspace when all of the spin-images from all of the models are taken together. This *library eigenspace* is spanned by *library eigenvectors* or *library eigen-spin-images*. The significant library eigenvectors describe the best directions for differentiating between models when the spin-images from different models are separated. Projecting each spin-image onto the significant library eigen-spin-images results in tuples for each spin-image that can be used for differentiating between models. Figure 6-9 shows a pictorial description of the difference between single model spin-image compression and library spin-image compression.

Using PCA to differentiate among spin-images from different models assumes that spin-images from different models come from separated clusters in spin-image space. In reality, this assumption may not be true because different models can have similar shape and, consequently, similar spin-images. In practice, we found that library eigen-spin-images generated using PCA were sufficient for differentiating between models in a library of 20 objects (see Chapter 7). If more discrimination between models is needed, perhaps due to an increase of the number of models in the model library, then a possibly more discriminating measure, e.g., Fisher Linear Discriminant [4][32], can be used.

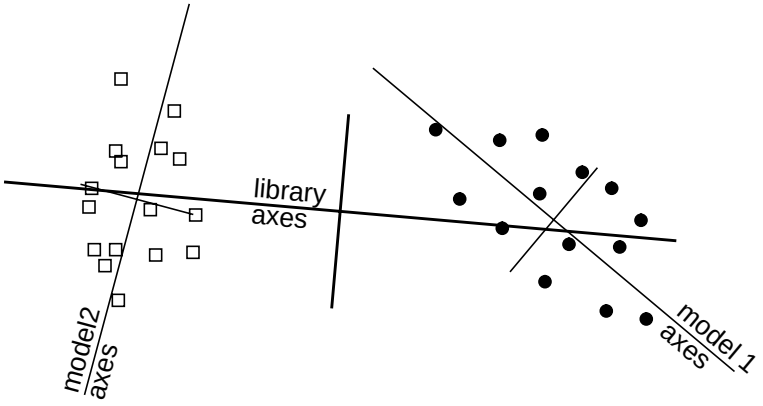


Figure 6-9: Difference between model compression and library compression. Model compression is used for discrimination between points belonging to the same model. Library compression is used to discriminate between points coming from different models.

6.4.1 Equal Weighting of Models in Library Compression

Constructing library eigen-spin-images is similar to constructing model eigen-spin-images. Once some additional steps are taken to ensure that spin-images from different model can be compared and that all models are given equal weight in library eigen-spin-image construction, spin-images from all of the models are placed in an array. A covariance matrix is computed from the array and then its eigenvectors are determined. These eigenvectors correspond to the best directions for differentiating between models.

As in the case of the model compression, care must be taken to ensure that the resolution of all of the models in the model library are the same resolution. Furthermore, the scene resolution should be similar to the resolution of the models. This will guarantee that *s*-tuples from matching spin-images have the same magnitude.

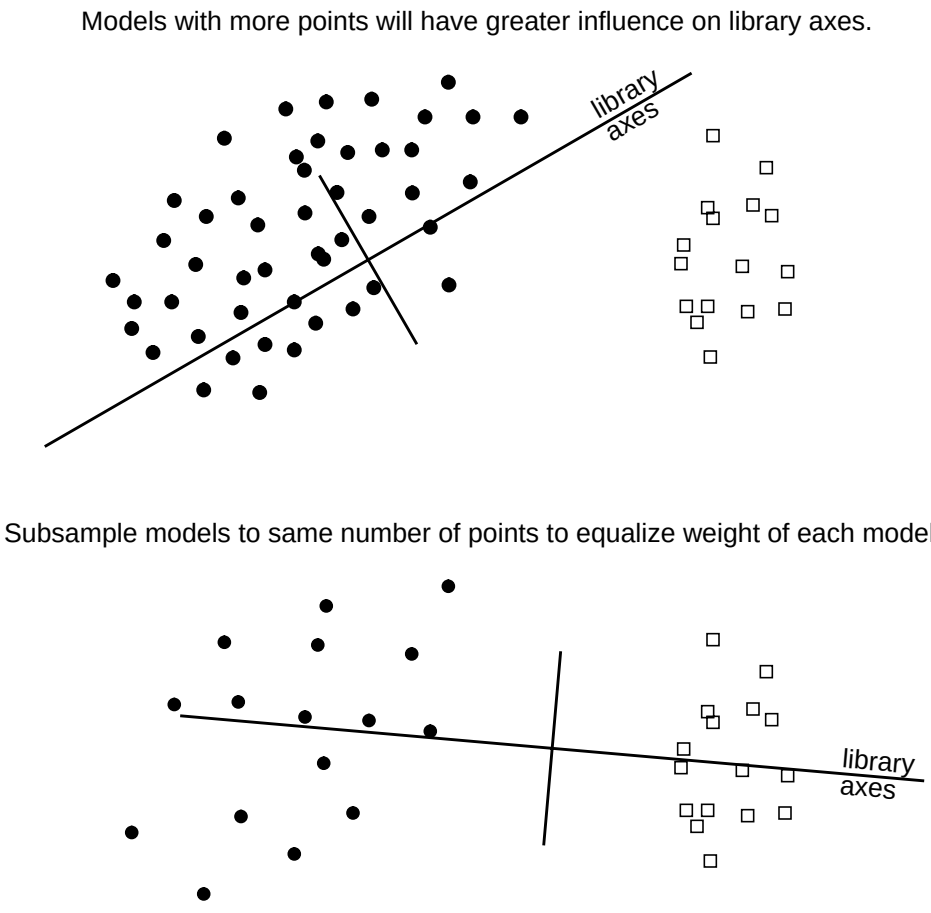


Figure 6-10: Pictorial description of problem with surface area differences on library eigen-spin-image computation

Since spin-images from multiple models will be compared to determine the library eigen-spin-image, the spin-image parameters (dimension, bin size, angle threshold) must be the same for all of the models. Comparison of spin-images is meaningful only when the spin-images are generated using the same spin-image parameters.

Each spin-image has an equal weight when determining the principal directions of a set of spin-images. This poses a problem when computing the library eigen-spin-images. Since all of the models are the same resolution, models of larger surface area will have more vertices and, consequently, more spin-images. If the spin-images from all of the vertices on all of the models are used to compute the library eigen-spin-images, then models with larger surface area will have more weight when determining the library eigen-spin-images. All models should be given equal weight, so that the library eigen-spin-images are optimal for differentiation between models. If a model is given more weight than the others, then the eigen-spin-images computed will begin to contain components that differentiate between the model vertices of the model with more weight in addition to continuing components for differentiating between models. Figure 6-10 shows a pictorial illustration of this problem.

A possible solution to the problem of unequal weights between the models when determining library eigen-spin-image compression is to make sure that all of the models have the same number of vertices. This can be accomplished by simplifying the model meshes using the algorithm described in Appendix A. There is a fundamental problem with this approach: simplification changes the resolution of the meshes. If all meshes are simplified to a fixed number of points, but all the models have different surface area, then all of the model meshes will have different resolutions. Since similar resolutions is a necessity for good s-tuple matching, this option is not feasible.

Another option is to compute the spin-images for all vertices on the model mesh and then pick a fixed number of spin-images from each model to be used in the determination of the library eigen-spin-images. This solution does not have the problems with resolution that the previous solution has because the spin-images are all computed for models of the same resolution. The difficulty with this solution is the choice of spin-images. The spin-images must be chosen so that they adequately convey the distribution of spin-images for each model. Random or uniform sampling of the spin-image indices does not guarantee coverage of the distribution of

spin-images, so some other method must be devised.

A better solution is a combination of the above two methods; simplification is used to select a set of vertices that will produce spin-images that adequately describe the spin-images of the entire model. First, using the algorithm in Appendix A, each model mesh M is simplified to a new model mesh M' with a fixed number of vertices. Next, for each vertex v' in M', the closest vertex v in M is determined. This is done efficiently using a kD-tree for M. Next, p, the closest point to v' on M is determined by projecting v' onto each face adjacent to v and setting p to the closest projection point. The normal to p is set to the normal of the face that p lies on, creating an oriented point. Since the vertices in M' are produced by a simplification algorithm that controls mesh resolution, when projected onto M, they will produce a uniform sampling of the surface of the model. Next, a spin-image is created for each oriented point p using the vertices in M. Since the surface of M is uniformly sampled, the spin-images produced in this way will accurately describe the distribution of spin-images for M. This solution works because it creates a fixed number of spin-images that adequately conveys the distribution of spin-images for each model. Furthermore, each spin-image is created from models of fixed resolution. Figure 6-11 shows a model and its subsampled mesh.

A possible alternative to our resampling procedure is to use weighted PCA. When computing library eigen-spin-images, the spin-images from each model can be weighted based on the number of vertices in the model. Spin-images from a model with many vertices would have a small weight and spin-images from models with a small number of vertices would have a large

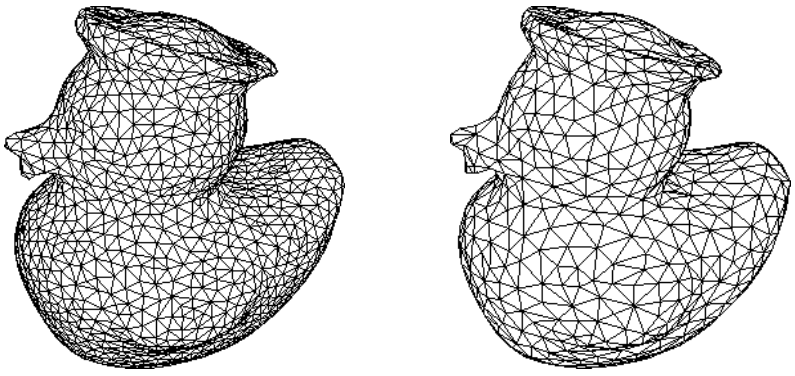


Figure 6-11: Sub-sampling of a mesh for equal weighting of models when calculating library eigen-spin-images.

weight. Since this approach to equalizing the contribution of each model to library spin-image generation will produce results comparable to our resampling approach, we did not implement it.

6.4.2 Library Eigen-Spin-Images

Given the preceding precautions, the library eigen-spin-images and library *t*-tuples can be created. Suppose there are *M* models M_j each with N_j spin-images of size *D* in the model library. Suppose that the fixed number of spin-images for each model is *F* and that the spin-images created by uniformly sampling M_j are $\boldsymbol{z_{ij}}$. The mean of these spin-images, call them uniform spin-images, for all models is

$$\bar{\boldsymbol{z}} = \sum_{j=1}^{M} \sum_{i=1}^{F} \boldsymbol{z_{ij}}. \tag{6.13}$$

Subtracting the mean of the uniform spin-images from each uniform spin-image makes the principal directions computed by PCA more effective for describing the variance between models.

$$\hat{\boldsymbol{z}}_{\boldsymbol{ij}} = \boldsymbol{z_{ij}} - \bar{\boldsymbol{z}} \tag{6.14}$$

The mean-subtracted stack of spin-images from the all the models can be represented as an *DxMN* matrix with each column of the matrix being a mean-subtracted spin-image

$$S^l = \left[\hat{\boldsymbol{z}}_{11}\ \hat{\boldsymbol{z}}_{12}\ \ldots\ \hat{\boldsymbol{z}}_{1F}\ \hat{\boldsymbol{z}}_{21}\ \ldots\ \hat{\boldsymbol{z}}_{2F}\ \ldots\ \hat{\boldsymbol{z}}_{M1}\ \ldots\ \hat{\boldsymbol{z}}_{MF}\right]. \tag{6.15}$$

The covariance of the spin-images in the model is the *DxD* matrix *C* given by

$$C^l = S^l (S^l)^T. \tag{6.16}$$

The eigenvectors of *C* are then computed by solving the eigenvector problem

$$\lambda_i^l \boldsymbol{e_i^l} = C^l \boldsymbol{e_i^l}. \tag{6.17}$$

Since the dimension of the spin-images is not too large (~100), the standard **`jacobi`** algorithm from Numerical Recipes in C [71]is used to determine the eigenvectors $\boldsymbol{e_j^l}$ and eigenvalues λ_i^l

of C^l. Since the eigenvectors of C^l can be considered as spin-images, they will be called library eigen-spin-images.

Figure 6-12 Shows a linear and logarithmic plot of the eigenvalues for the library eigen-spin-images made from the 10 toy models shown in Figure 4-12. After the first 20, the eigenvalues become negligible. This indicates that the library spin-images live in a small dimensional subspace of the eigenspace and can be accurately described by projections onto a small number of library eigen-spin-images. The method for determining where to cutoff the projection dimension is the same as for determination of the cut off of model spin-image dimensions as discussed in Section 6.2.3.

Once the library projection dimension *t* has been fixed, the library *t*-tuples for each model are determined as projections onto the library eigen-spin-images. For each model M_j, create a spin-image $\boldsymbol{x_{ij}}$ for each on of its vertices v_i, using the library spin-image parameters. Next, subtract the mean of the library spin-images to get the mean-subtracted model spin-image $\hat{\boldsymbol{x}}_{\boldsymbol{ij}}$. The library *t*-tuple representing the model spin-image is then the projection of the mean-subtracted model spin-image onto the first *t* library eigen-spin-images

$$\boldsymbol{r_{ij}} = (\hat{\boldsymbol{x}}_{\boldsymbol{j}}\boldsymbol{e}_{\boldsymbol{1}}^{\boldsymbol{l}}, \hat{\boldsymbol{x}}_{\boldsymbol{j}}\boldsymbol{e}_{\boldsymbol{2}}^{\boldsymbol{l}}, \ldots, \hat{\boldsymbol{x}}_{\boldsymbol{j}}\boldsymbol{e}_{\boldsymbol{s}}^{\boldsymbol{l}}). \tag{6.18}$$

The model library t-tuples are then stored in a single efficient closest point search structure described in Section 6.3.3.

Once the library tuple has been created for each model point, determination of the best matching *model* to a scene point is very similar to finding the best matching *model point* to a scene point. Given a scene oriented point, its spin-image is created using the scene surface and the library spin-image parameters. The mean of the library spin-images is then subtracted from the scene spin-image which is then projected onto the *t* most significant library scene eigen-spin-images to get a scene *t*-tuple. The scene *t*-tuple is then used as query point into the library efficient closest search structure which returns the model and model vertex associated with the closest tuple in the library eigen-space. The next section explains four different model-based recognition algorithms that can be used, based on different combinations of model spin-image compression.

6.5 Recognition Algorithms

By combining spin-image matching with no compression, model compression and library compression, four different spin-image matching algorithms have been developed. The algorithms differ only in how they match spin-images. Once spin-images are matched, the surface matching engine determines the final recognition results in all cases. In this section, the four different algorithms are described, along with their expected performance. In the next chapter, the algorithms are compared experimentally in terms of recognition rates and running times.

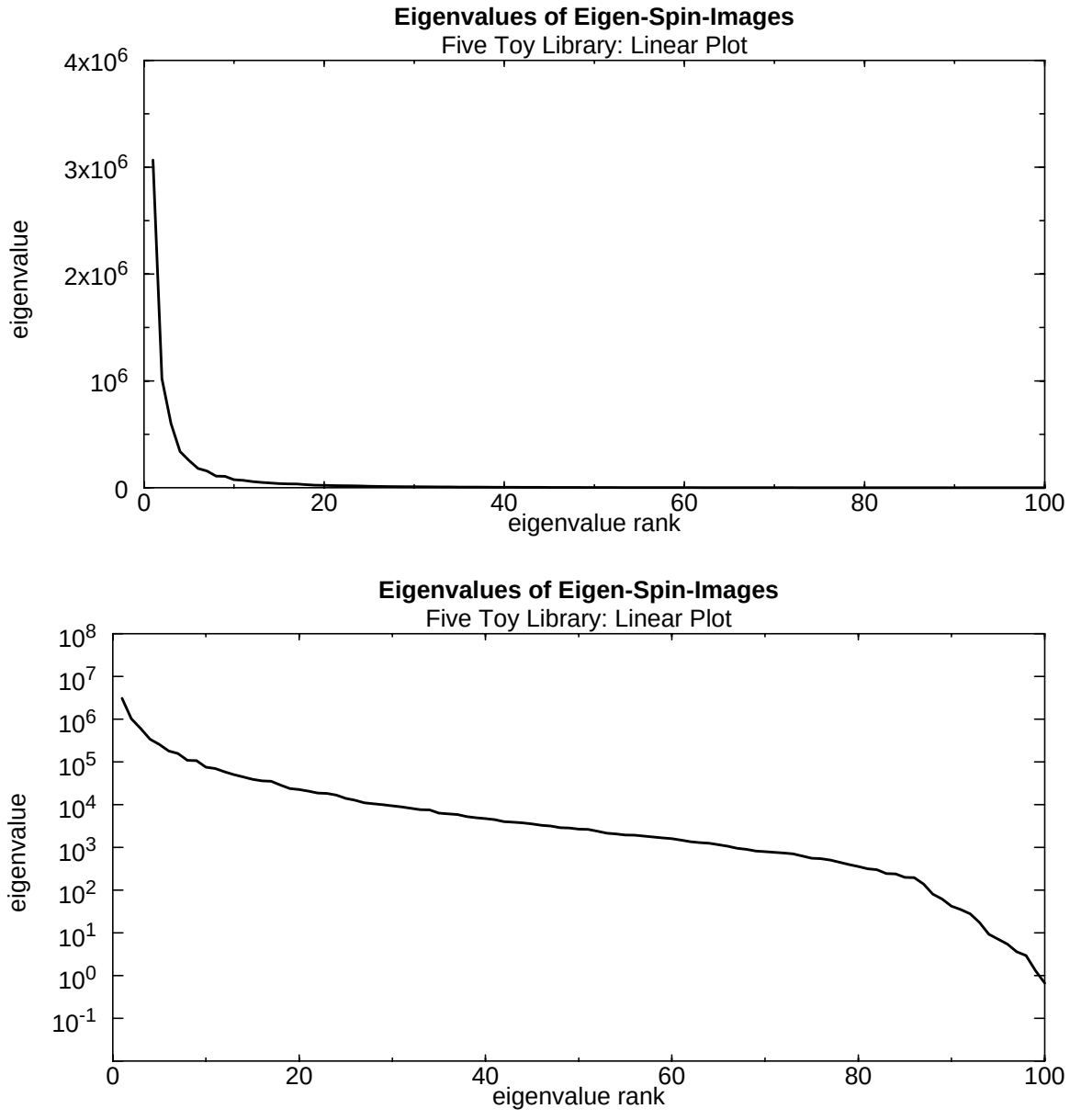


Figure 6-12: Eigenvalues for library spin-images

6.5.1 Matching With No Compression: MA1

Recognition with no compression is the algorithm presented in Chapter 5. Since no compression or efficient closest point search is used, this algorithm is the least efficient of the four. As explained in Section 3.6, the total complexity of matching a single model to a scene is $O(S^2D)$. Matching M models in a model library without compression requires matching each model separately, so the total complexity of MA1 is $O(MS^2D)$.

Since the model spin-images are not compressed, no information is removed before comparing

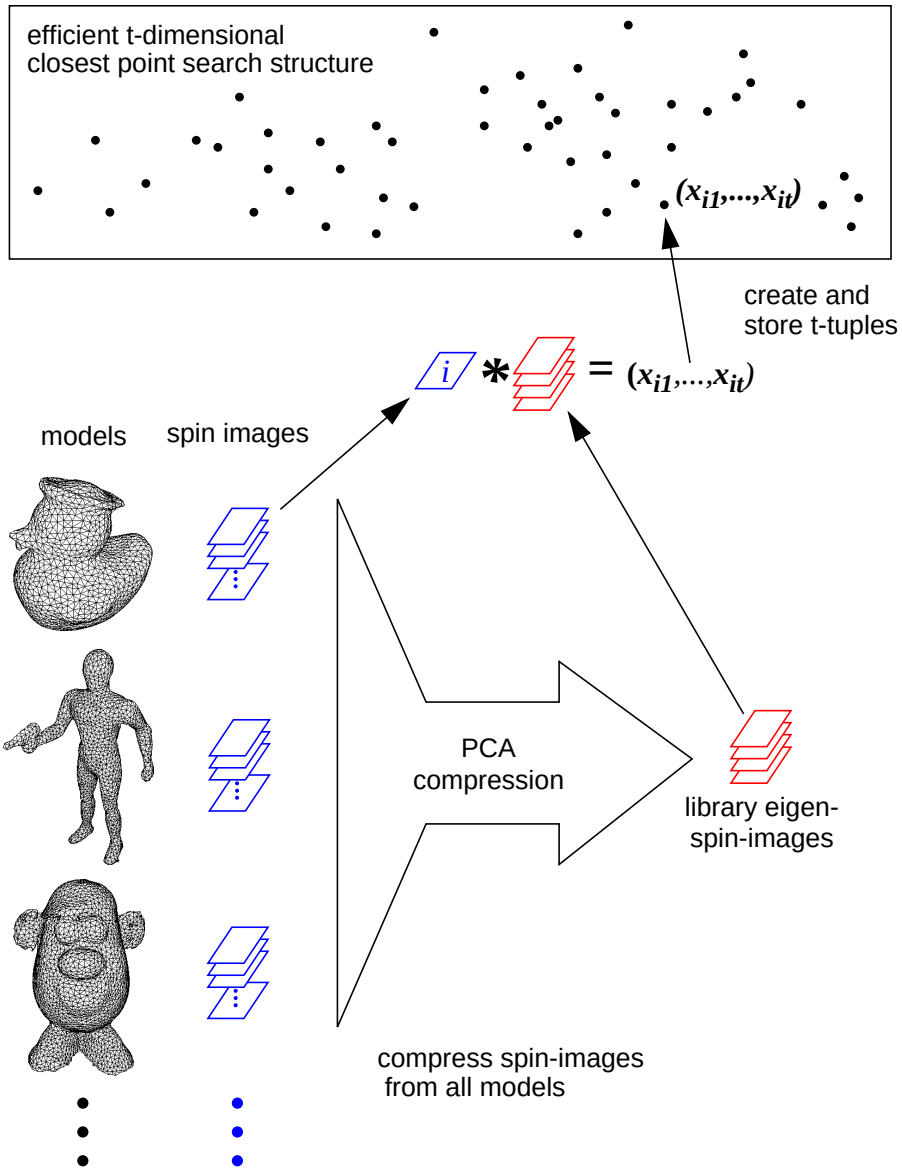


Figure 6-13: Model library compression. Spin-images from multiple models are compressed into a single set of eigen-spin-images.

spin-images. Furthermore, each scene spin-image is compared to all spin-images from all models, and point matches are ranked based on similarity measure. Similarity measure is a stronger indicator of match correctness than the l_2 norm because it measures correlation as well as overlap between images. Ranking all matches using similarity measure will guarantee that the best match between model and scene will be found using the best comparison function. Therefore, it is expected that, in this case, the correctness of the spin-image matching will be the best of all of the algorithms.

6.5.2 Matching with Model Compression: MA2

Matching with model compression is the matching algorithm described in Section 6.3. In this algorithm, spin-images from a single model are compressed into model s-tuples. Matching spin-images is reduced to closest point search between *s*-tuples, so, in calculation of the computational complexity, spin-image dimension D is replaced by the dimension *s* of the *s*-tuple According to Nene and Nayar [67], their algorithm for finding closest points is linear in the number of points in the search structure and independent of the dimension of the points. Therefore, comparing a scene s-tuple to a model is *O(N*), not *O(ND)* as in MA1. This algorithm, like the previous one, must search through all of the models, so the total computational complexity of matching a single spin-image is *O(MN)*. This improves the spin-image matching complexity from $O(MS^2D)$ for MA1 to $O(MS^2)$ for MA2. Besides having a better computational complexity, the multiplicative constant for MA2 will be smaller than the constant for MA1 because correlation coefficients do not have to be computed for spin-image matching.

The cost of the improvement in computational complexity is less accurate matching. To match spin-images, MA2 uses the l_1 and l_2 distance which is less accurate than similarity measure. The reduction in information due to compression may also degrade matching accuracy. However, the effect of the difference in distribution of vertices over the model and scene that will be present in the uncompressed spin-images may not be represented in compressed spin-images. Therefore, in some cases, matching compressed spin-images may yield more accurate matching results than matching uncompressed spin-images.

6.5.3 Matching with Model Compression and Library Compression: MA3

MA3 adds library compression to model compression to obtain a more efficient matching al-

gorithm. This algorithm is along the lines of the algorithm presented by Murase and Nayar [65] for appearance based recognition. The algorithm works as follows: Before recognition occurs, *s*-tuples are made for each model spin-image using model compression, and *t*-tuples are made for each model spin-image using library compression. At recognition time, a random fraction of scene points is selected. For each scene point, a scene spin-image is created using the library spin-image parameters. This scene spin-image is projected onto the library eigen-spin-images to get a scene *t*-tuple. The closest matching library *t*-tuple is then returned from the library closest point search structure. The model associated with this t-tuple is considered the best matching model to the scene spin-image.

Next, a scene spin-image for the current scene point is created using the spin-image parameters of the best matching model. This is followed by construction of a scene *s*-tuple by projection onto the best model eigen-spin-images. Close matching model *s*-tuples are then returned from the model closest point search structure. These *s*-tuples indicates possible matching model points from which a point matches are created and passed to the surface matching engine.

The attraction of MA3 is that it can determine the correct model matching a scene point, from in a library of models, much more quickly than MA2. Since there are MN *t*-tuples in the library closest point search structure, finding the best model matching a scene point is $O(MN)$. This is the same computational complexity as MA2, but the constant multiplicative time in the case of MA3 should be much less than in the case of MA2. Once the best model is determined, finding the best matching model point is $O(N)$ using the model closest point search structure. This results in a complexity of $O(MN) + O(N) = O(MN)$ for matching each scene spin-image. This results in a total matching complexity of $O(MS^2)$ which is the same as the complexity of MA2. However, MA3 will have a smaller multiplicative constant than MA2.

MA3 also requires more storage than MA2. For each model spin-image, a model s-tuple and a library t-tuple must be stored. In addition, the library-eigen-spin-images must be stored. It is conceivable that storing two sets of tuples for each model will not result in any additional storage if a balance can be met between model and library tuple dimensions. In MA2, model tuples are used for model determination and model point localization. In MA3, model tuples are used for model point localization and library tuples are used for model determination. Since the model tuples in MA2 are performing both localization and recognition, they may need to be of

a higher dimension than if they are being used for localization alone. It may be the case that fewer dimensions are needed for the model tuples in MA3 because they are being used only for localization. By reducing the dimension of the model tuples, room is made for library tuples. More investigation into these trade-offs will be done in the future.

It is expected that MA3 will have less match accuracy than MA2 because each model is not checked for matches. If the wrong model is picked in the first stage of the algorithm, then model point matching cannot possibly determine the correct match. This could possibly be alleviated by returning the best few models in the library matching stage, but the result would be an increase in running time linear in the number of matches returned, possibly negating the benefits of the library search stage.

6.5.4 Matching with Library Compression: MA4

Matching with library compression skips model compression altogether and uses library tuples for recognition and localization. The MA4 algorithm is the same as the MA3 algorithm up to the point of matching the scene library tuple to the model library tuples. Instead of returning only the best matching model tuple from the library closest point search structure, the best few (within ε) model tuples are returned. From each model tuple, the matching model and model point can be determined. The matching models and model points are matched to the scene point and these point matches are passed to the surface matching engine.

The attraction of MA4 is that it can efficiently search for the correct model and model point at the same time. Since there are MN t-tuples in the library closest point search structure, finding the best model and model points is O(MN). This complexity is still linear in the number of models, but as in MA3, the constant multiplicative time is small. Since MA4 skips the step that searches for the matching model vertex needed in MA3, it is the most efficient algorithm of the four, although its total algorithmic complexity is still $O(MS^2)$. MA4 also has fewer storage requirements than MA3 since only one set of tuples needs to be stored for each model spin-image. However, the library tuples used in MA4 may need more dimensions than those in MA3 in order to effectively perform localization and recognition.

MA4 will be the least accurate matching algorithm because recognition and localization are performed in the same step. In MA4, tuples created for recognition and localization come from

the library eigenspace which is used to differentiate between models and is not as good at localization. However, if the spin-images from all models live in a similar subspace of the spin-image space then the library eigenspace will be useful for recognition and localization.

The next chapter shows results from experiments using the four matching algorithms. The algorithms are compared based on running times for different numbers of models and recognition rate for a fixed number of models. Our experiments validate that matching with compression is much more efficient that matching without compression. Furthermore, compression does not drastically reduce the accuracy of spin-image matching.

Chapter 7

Recognition Experiments

In this chapter, we demonstrate the effectiveness of spin-images for recognition using qualitative and quantitative measures. First, we show qualitative views of recognized and localized models superimposed on the scene data. These results demonstrate the simultaneous recognition of multiple objects in many different scenes using many different models and model libraries. Next, by performing hundreds of recognition experiments, we quantitatively evaluate the effect of clutter and occlusion on recognition using compressed and uncompressed spin-images. We conclude that recognition success is robust to clutter, while increasing occlusion results in decreasing recognition success. Finally, we experimentally evaluate the running times and recognition rate as functions of the number of models in the model library for the four matching algorithms described in Chapter 6. Comparison of running times shows that compression results in an order of magnitude decrease in spin-image matching time.

7.1 Qualitative Recognition Results

Before a recognition experiment is run, the model library has to be built. The models we use in our recognition experiments were created using the object modeling algorithm described in Chapter 4. To accurately convey surface details, the object modeling algorithm creates meshes with a fine resolution. To increase matching speed, the fine version of each model is simplified to a coarser mesh with a resolution of 0.5 cm using the algorithm for control of mesh resolution described in Appendix A. Consequently, each model has the same mesh resolution. For the recognition experiments shown in this chapter, all of the model spin-images are created using the same spin-image generation parameters: bin-size of 0.5 cm, image width of 10, and support an-

gle of 60 degrees. These were chosen by combining the analysis of spin-image parameters of Chapter 9 and the clutter model from Chapter 5. By generating recognition results for a fixed set of parameters, we show that recognition does not rely on careful tweaking of parameters to achieve recognition for a wide variety of scenes.

To create model libraries, multiple model meshes along with their spin-images are combined. Based on the requirements of the matching algorithms, the spin-images may or may not be compressed with tuples stored in an efficient closest point search structure. For all of the results shown in this chapter, the model compression dimension is set to 20 ($s=20$) and the library compression dimension is also set to 20 ($t=20$). An image width of 10 corresponds to an image dimension of 100, so the model and library tuples compress the original spin-images by 80%. Although the model and library dimension can be set based on reconstruction level, as explained in Section 6.2.3, they were fixed at the same value of 20 so that all models and libraries have the same compression. This removes the influence of the amount of compression from the comparison of recognition results using different libraries.

When matching spin-images with compression, the closest point search threshold ε must be set. As explained in Section 6.3.3, we set ε to 1.5 times the median distance between tuples. Since the median distance between tuples will change depending on the libraries, the actual search distance will change from library to library. However, the user input of 1.5 remains fixed for all experiments and all libraries.

The scenes data was acquired by placing objects in front of a structured light range sensor and taking a range image using the setup described in Chapter 4. The scene data was then processed to remove isolated points, dangling edges and small patches. This topological filter was followed by mesh smoothing without shrinking [88] and mesh simplification (Appendix A) to change the scene mesh resolution to 0.5 cm. After processing, the scene surface is ready to be matched to the models in the selected model library.

The surface matching parameters were also fixed for all experiments. The geometric consistency threshold used to group correspondences (Section 3.3) was set to 0.25. The verification distance threshold used when verifying model transformations (Section 3.5) was set to 1.0 (2.0

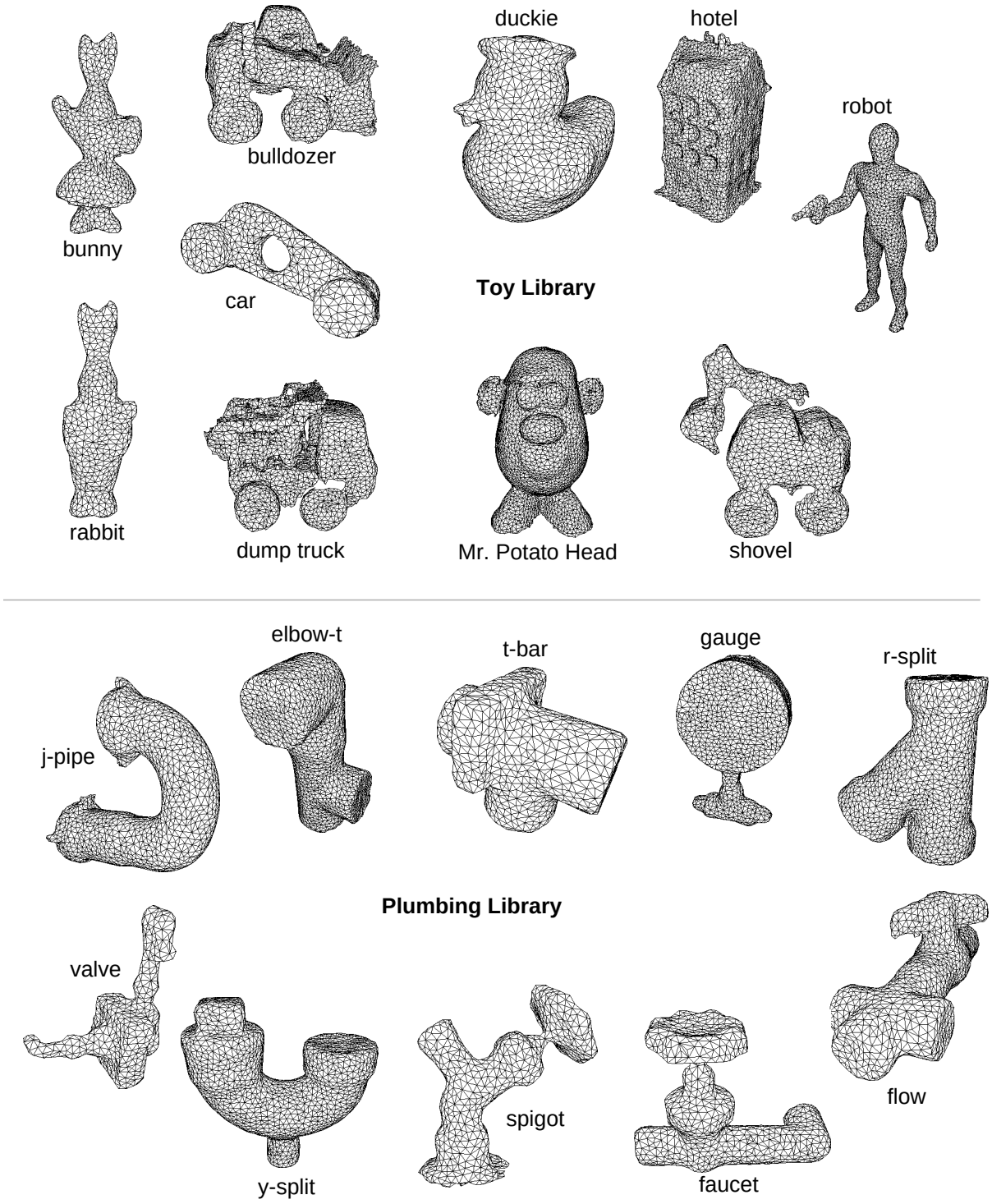


Figure 7-1: Mesh models used in model libraries. The Toy Library consists of 10 models of toys and the Plumbing Library consists of 10 models of pipes and valves. Each model was made using the modeling from range images algorithm described in Chapter 4. Altogether the 20 models make up the large model library.

times the model resolution). Finally, the matched surface area threshold for determining presence of the model in the scene (Section 3.6) was set to one tenth the model surface area. These parameters will vary based on the model, but the parameters set by the users will remain fixed.

7.1.1 Large Model Library: 20 Models

The large model library was created to demonstrate the effectiveness of recognition when the model library contains a large number of models. In the case of the Large Model Library, we use all twenty of the models shown in Figure 7-1. For correct recognition to occur using a large number of models, spin-images must be able to differentiate between many models and the many locations on the model. Furthermore, we would like recognition to work in the presence of clutter and occlusion and when spin-images are compressed and uncompressed. The scenes for testing the large model library were created by filling the field of view of the sensor with objects from the model library; the objects were essentially piled in front of the sensor. This created scenes where the models are very close together and therefore contained large amounts of clutter and occlusion.

Figure 7-2 shows the simultaneous recognition of seven models using spin-images with library compression (MA4). In the top right of the figure is shown the intensity image available from the range sensor. It shows a scene with objects that are closely packed together. In the top left is shown the scene intensity image with the position of recognized models superimposed as white dots. In the middle is shown a frontal 3-D view of the scene data, shown as wireframe mesh, and then the same view of the scene data with models superimposed as shaded surfaces. The bottom shows a top view of the scene and models. The rest of the qualitative recognition results in this chapter are laid out in the same format. From the three views, it is clear that the models are closely packed creating a very cluttered scene with occlusions. Because spin-image matching has been designed to be resistant to clutter and occlusion, our algorithm is able to recognize the seven most prominent objects in the scene with no incorrect recognitions. Furthermore, this result was achieved with the MA4 algorithm, which uses the highest level of compression.

Figure 7-3 shows the recognition of six models from the large model library using library compression (MA4) for spin-image matching. The six models are all correctly localized. Some

models are not recognized in the scene because insufficient scene data on the surfaces of the models exists.

Figure 7-4 shows the recognition of five of the models in the Large Model Library using model and library compression (MA3). Figure 7-5 shows the recognition of six models from the Large Model Library using library and model compression (MA3). The bunny and rabbit models were differentiated by the algorithm even though the visible surfaces on the two models have very similar shapes.

7.1.2 Plumbing Library: 10 Models

The next set of results shows the recognition of the models made of plumbing parts. These models are shown at the bottom of Figure 7-1. The Plumbing Library was designed to test recognition using spin-images on parametric objects of similar shape. It was also designed to demonstrate the added difficulty of recognizing objects containing symmetry; for the most part, the Plumbing Library comprises pipe-like, and hence symmetric, objects. Symmetry always affects object recognition. If symmetry is not handled during recognition, it can often cause errors in recognition. Since our algorithm does not account for symmetry, its effect will be noticeable in the recognition results.

To create the scenes, models from the Plumbing Library were placed on a table along with other objects that are not present in the Plumbing Library. The range data acquired contains model surfaces along with surfaces from other objects, so these recognition results demonstrate the ability of the algorithm to not only correctly match models to scene surfaces, but also to reject scene surfaces that do not match any of the models in the model library. Some portions of the scene are missing, because they were too dark to be imaged by the range sensor.

Figure 7-6 shows the recognition of four models from the Plumbing Library using model and library compression (MA3). All models present in the scene are correctly recognized; models not in the scene are not matched to the scene. This result demonstrates the ability of spin-images to represent global shape. Locally, all of the recognized models are similar because they are made from cylinders of roughly the same shape. Algorithms using local shape descriptions (for instance, curvature [14][28][36]) would have difficulty with this scene. However, because spin-images encode global shape, they can easily differentiate among the objects.

Figure 7-7 shows the recognition of four models from the Plumbing Library using model compression (MA2). Figure 7-8 shows the recognition of three models from the Plumbing Library using library compression (MA4). The spigot and faucet model both have handles of similar shape, but spin-image matching is still able to differentiate between the models.

All of the models in the Plumbing Library contain symmetry. Although symmetry can adversely affect recognition, it can be detected using spin-images. If it can be detected then appropriate changes to the recognition algorithm can be made to handle symmetry. Detection and use of symmetry in spin-image matching is discussed as part of future work in Chapter 10.

7.1.3 Toy Library: 10 Models

The last set of results comes from a library of ten models made from the models of toy objects shown at the top of Figure 7-1. The Toy Library was designed to test recognition using models of complex, free-form shape and varying sizes and intensity.

Figure 7-9 shows the recognition of seven of the ten models in the Toy Library using the spin-image matching without compression (MA1). This result demonstrates the recognition of a large number of models of varying sizes and intensities in a cluttered scene with occlusion.

Figure 7-10 shows the recognition of five models from the Toy Library using the library spin-image compression for model identification and localization (MA4). This result demonstrates that the library tuples are sufficient for differentiating between models that are similar. The bunny model and the rabbit model have similar shape as do the shovel model and dump truck model. Furthermore, the stacking of the models introduces significant clutter into the matching process. Acquiring the scene data in this example was difficult because the large variation in surface brightness of the models in the scene. The bunny and rabbit models are almost white, while the dump truck, hotel and shovel models contain both light and dark areas. To capture shape information on the dark portions of the models, we had to increase the aperture of the camera. The larger aperture caused some saturation of the structured light patterns on the bright portions of the models, resulting in noisy and incomplete shape information. In spite of this, the models were still correctly recognized in the scene, demonstrating the ability of spin-images to handle noise.

Figure 7-11 demonstrates the recognition of three similar models: the dump truck, shovel and

bulldozer models using library compression (MA4). During scene data acquisition, the dump truck and shovel objects were placed on the floor and the bulldozer model was held in mid-air by the human operator. This result demonstrates the recognition of three similarly shaped objects using library compression for recognition and localization. The problem with variations in model brightness that were an issue in the scene shown in Figure 7-10 is an issue here as well. The models of the dump truck, bulldozer and shovel are inaccurate because of the difficulty in acquiring range data for model building when the scene intensity varies greatly. This effect also corrupts the scene data acquired, which is indicated by the horizontal stripes in the scene data on the cab of the dump truck. In spite of noisy models and scene data, spin-image matching is able to recognize and localize the models.

The results presented show that all of the algorithms can be used to recognize multiple objects using a library containing many models in scenes containing clutter and occlusion. In Section 7.3, we show how the algorithms compare in terms of running times and recognition rates as the number of models in the libraries is increased. In the next section we present an experimental evaluation of the effect of clutter and occlusion on recognition using single model libraries.

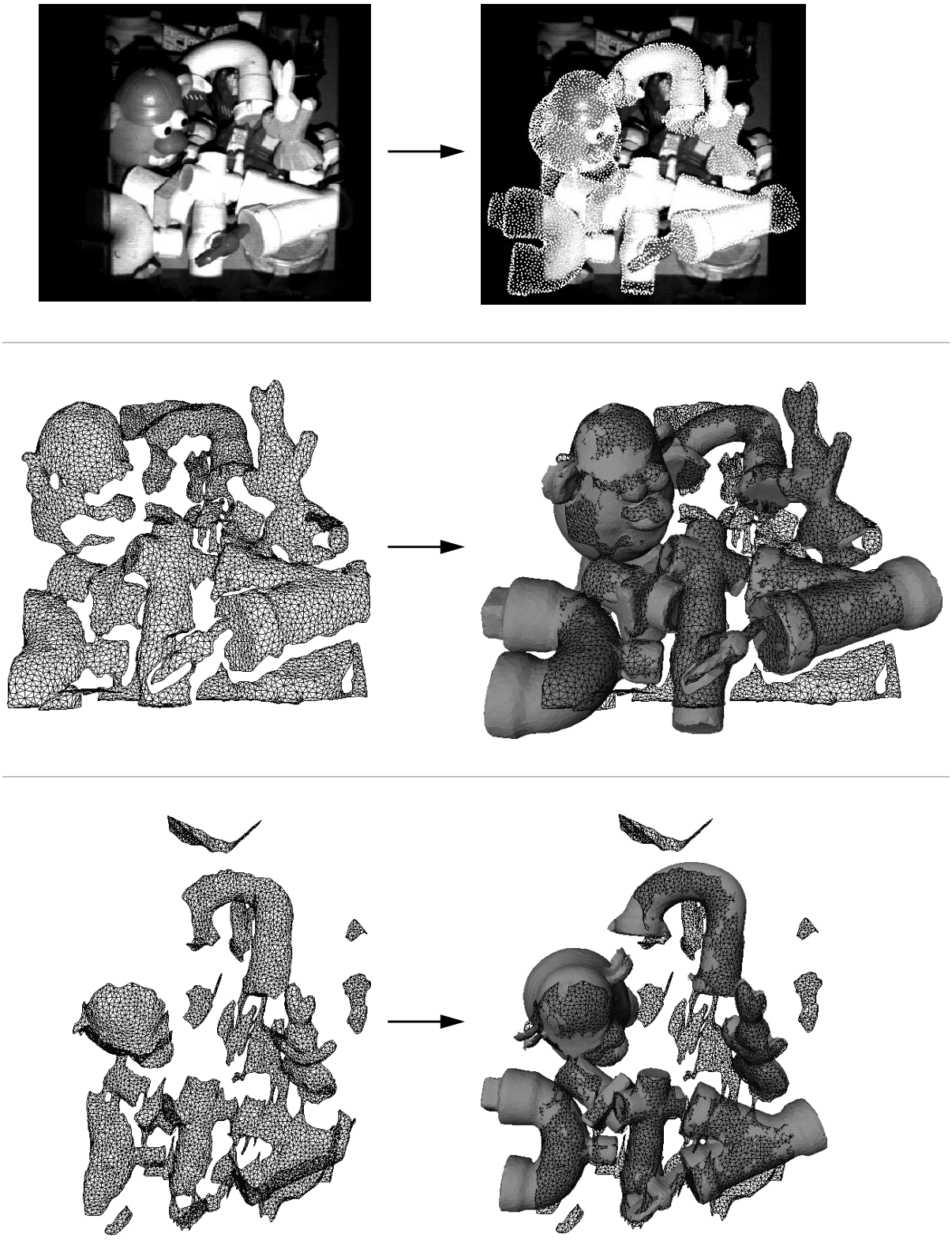


Figure 7-2: Large Model Library recognition of 7 models using library compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

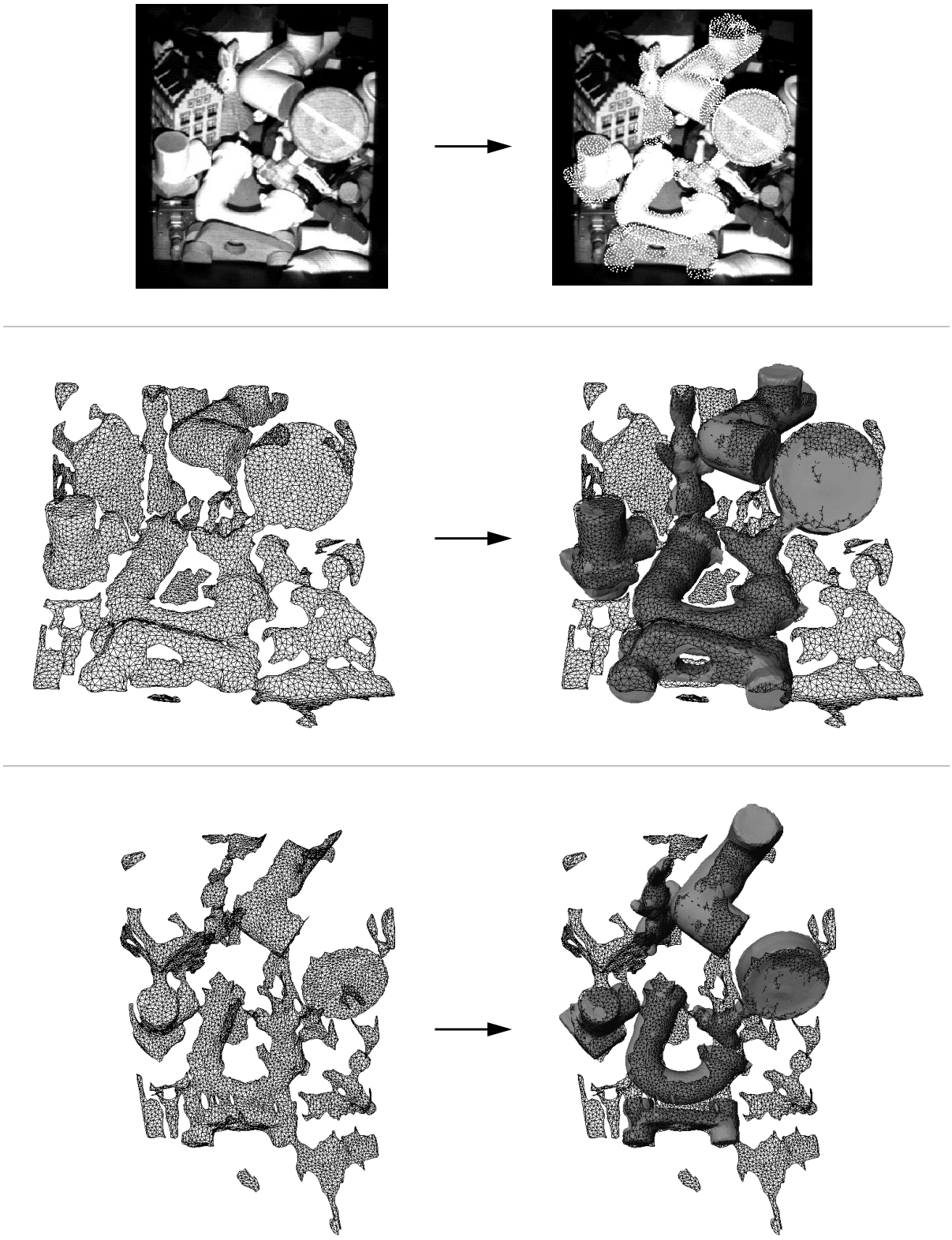


Figure 7-3: Large Model Library recognition of 5 models using library compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

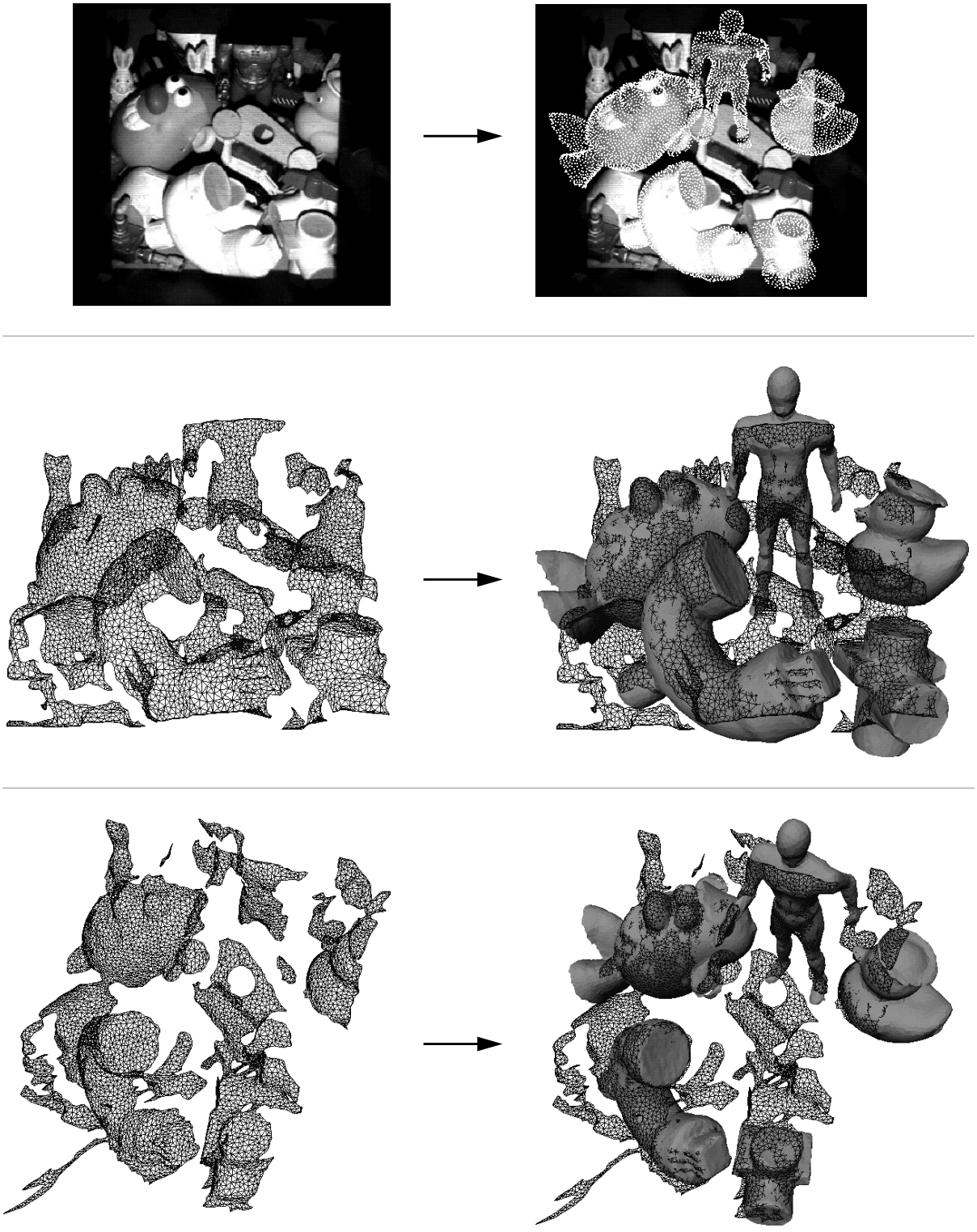


Figure 7-4: Large Model Library recognition of 6 models using library and model compression (MA3). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

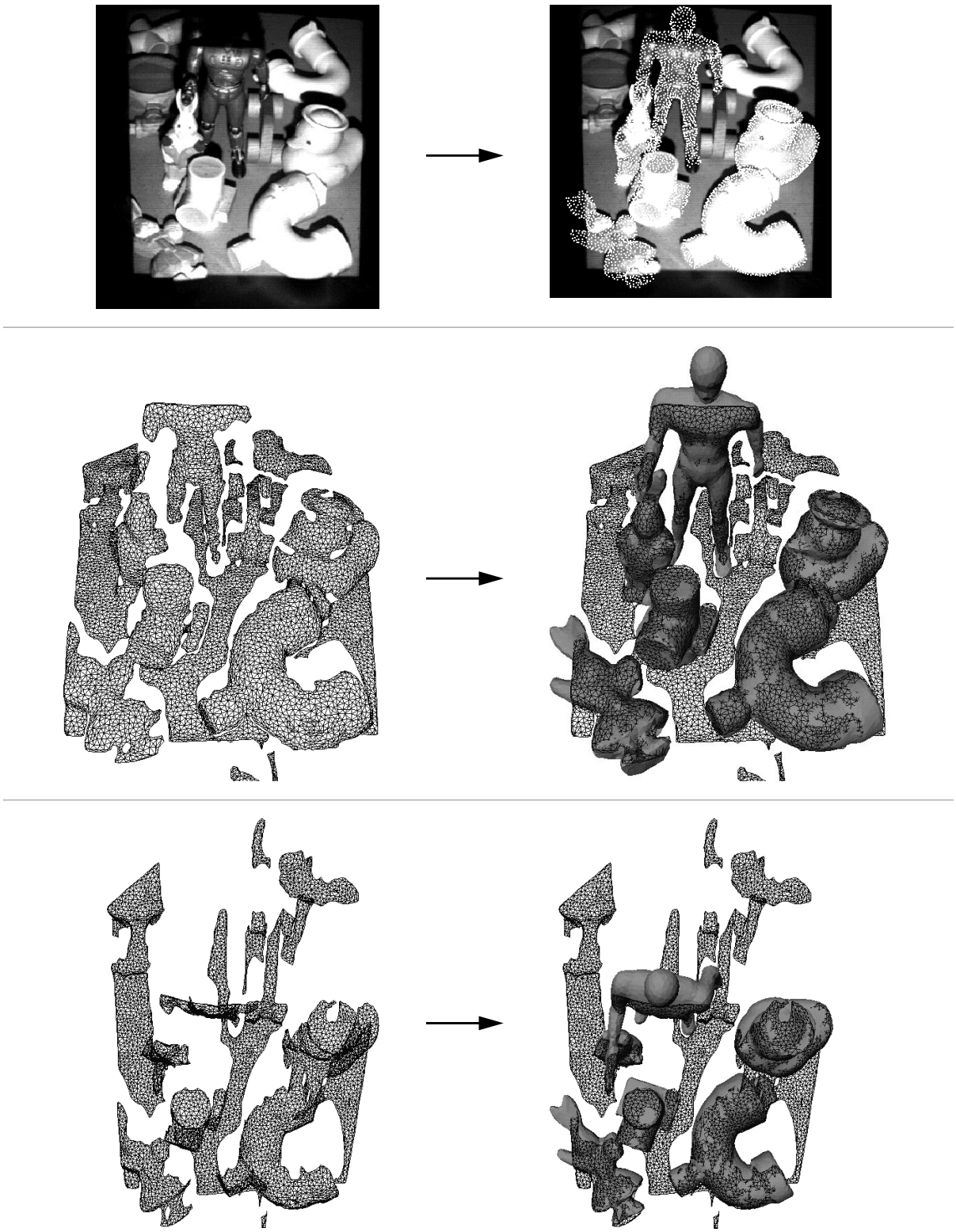


Figure 7-5: Large Model Library recognition of 6 models using library and model compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

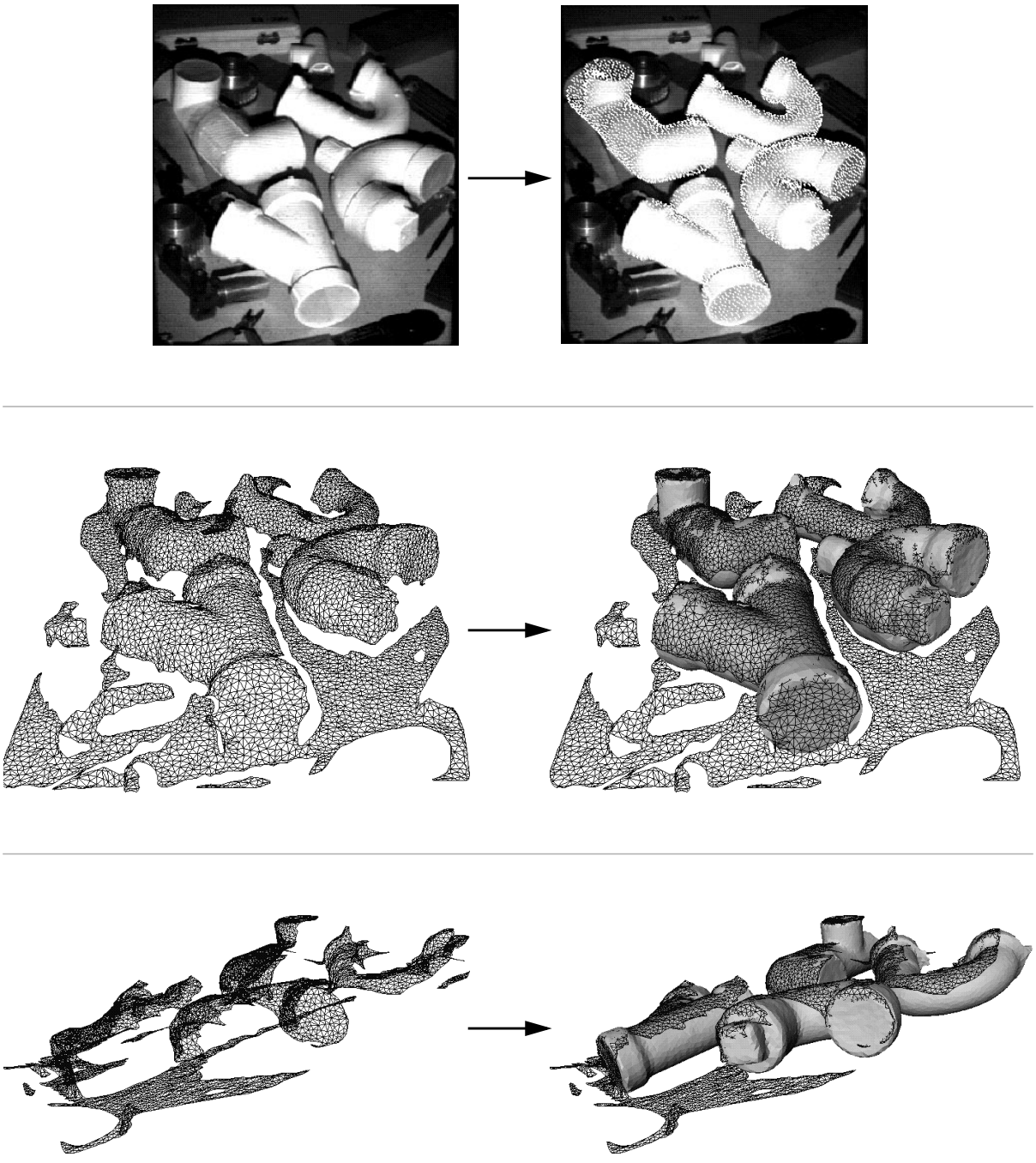


Figure 7-6: Plumbing Library recognition of 4 models using library and model compression (MA3). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a side view of scene mesh and shaded models (bottom). This result demonstrates the simultaneous recognition of models that are similar. All recognized models are made of cylindrical cross-sections of roughly the same radius.

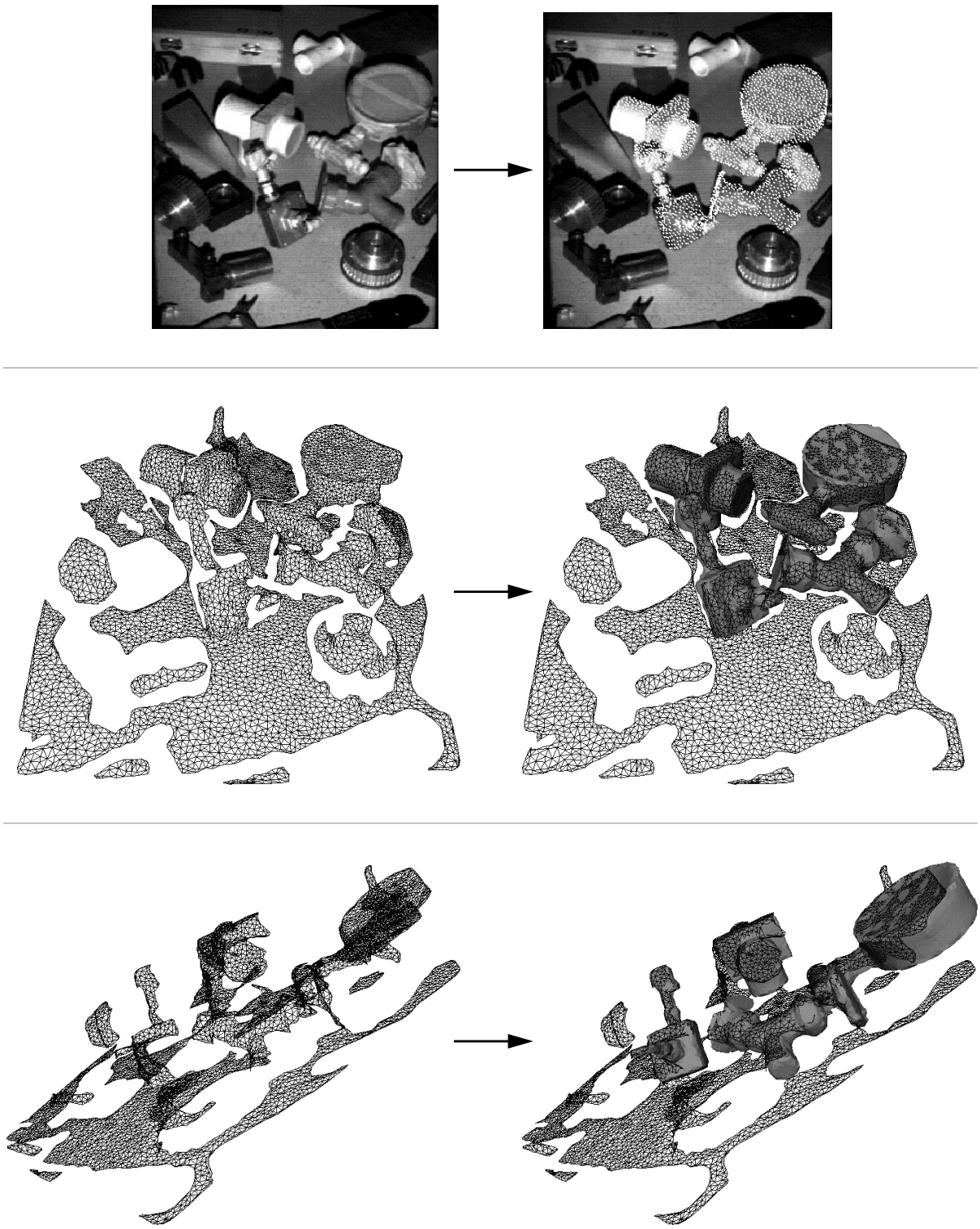


Figure 7-7: Plumbing Library recognition of 4 models using model compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a side view of scene mesh and shaded models (bottom).

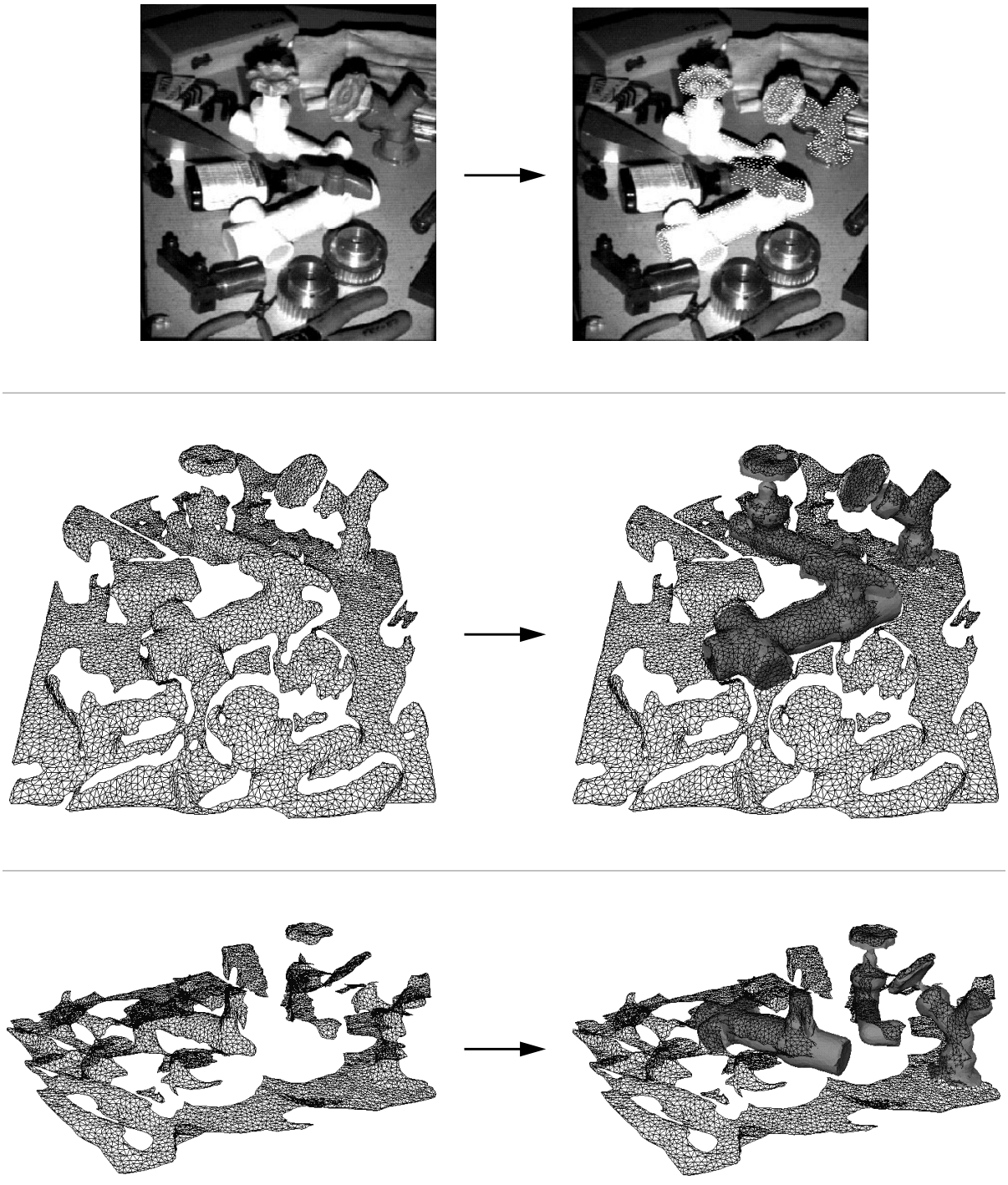


Figure 7-8: Plumbing Library recognition of 3 models using library compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a side view of scene mesh and shaded models (bottom).

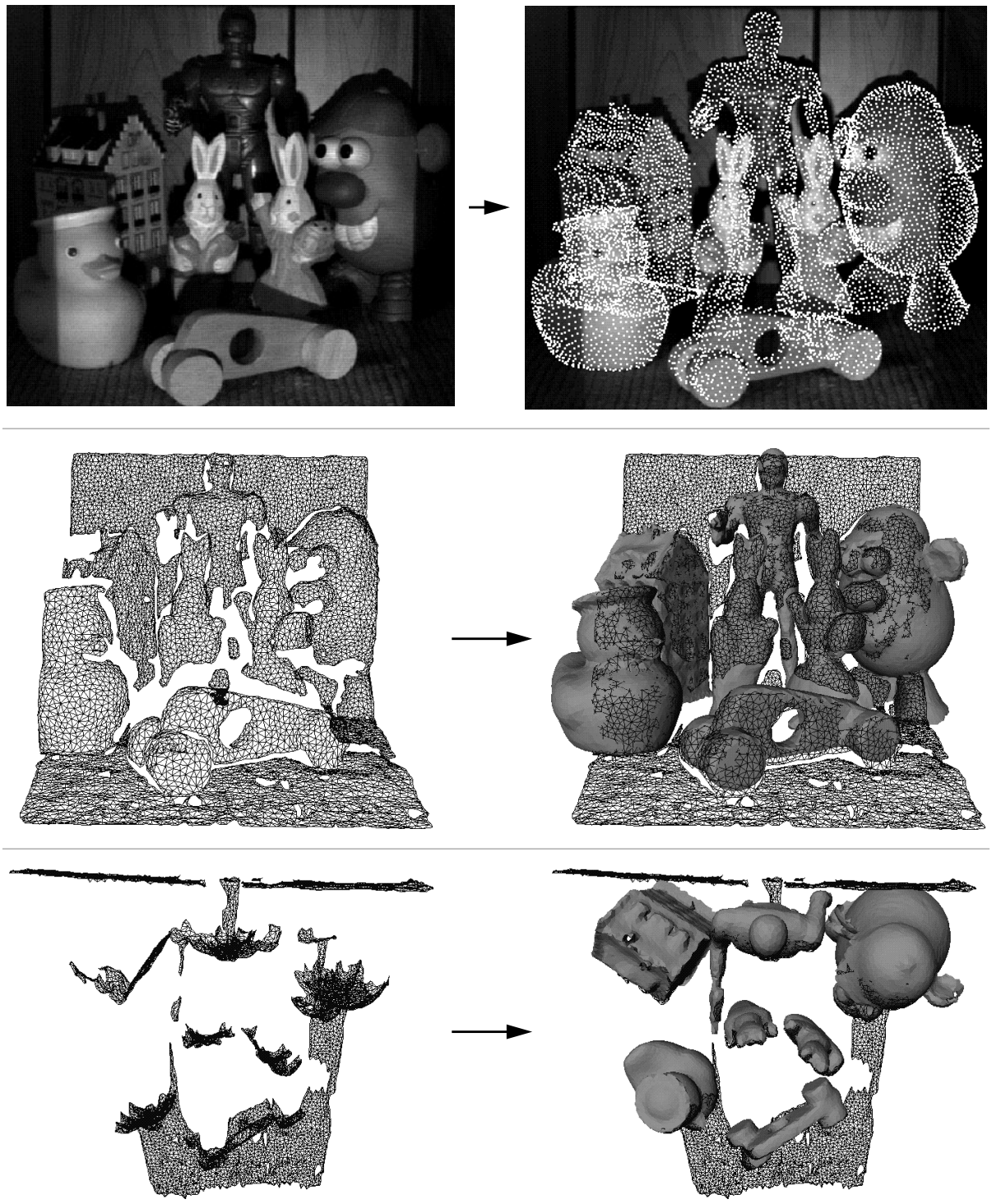


Figure 7-9: Toy Library recognition of 7 models using library and model compression (MA3). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

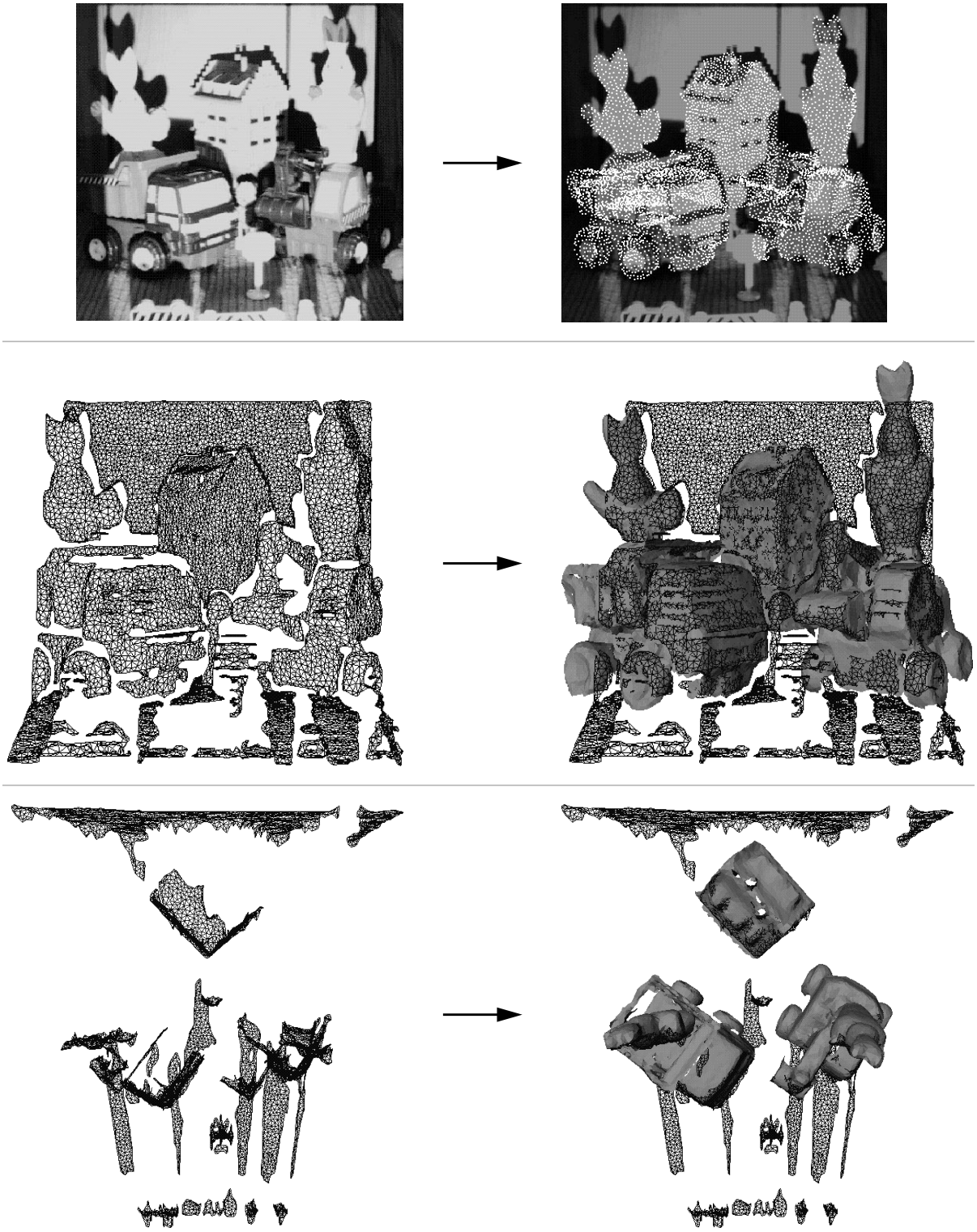


Figure 7-10: Toy Library recognition of 5 models using library compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

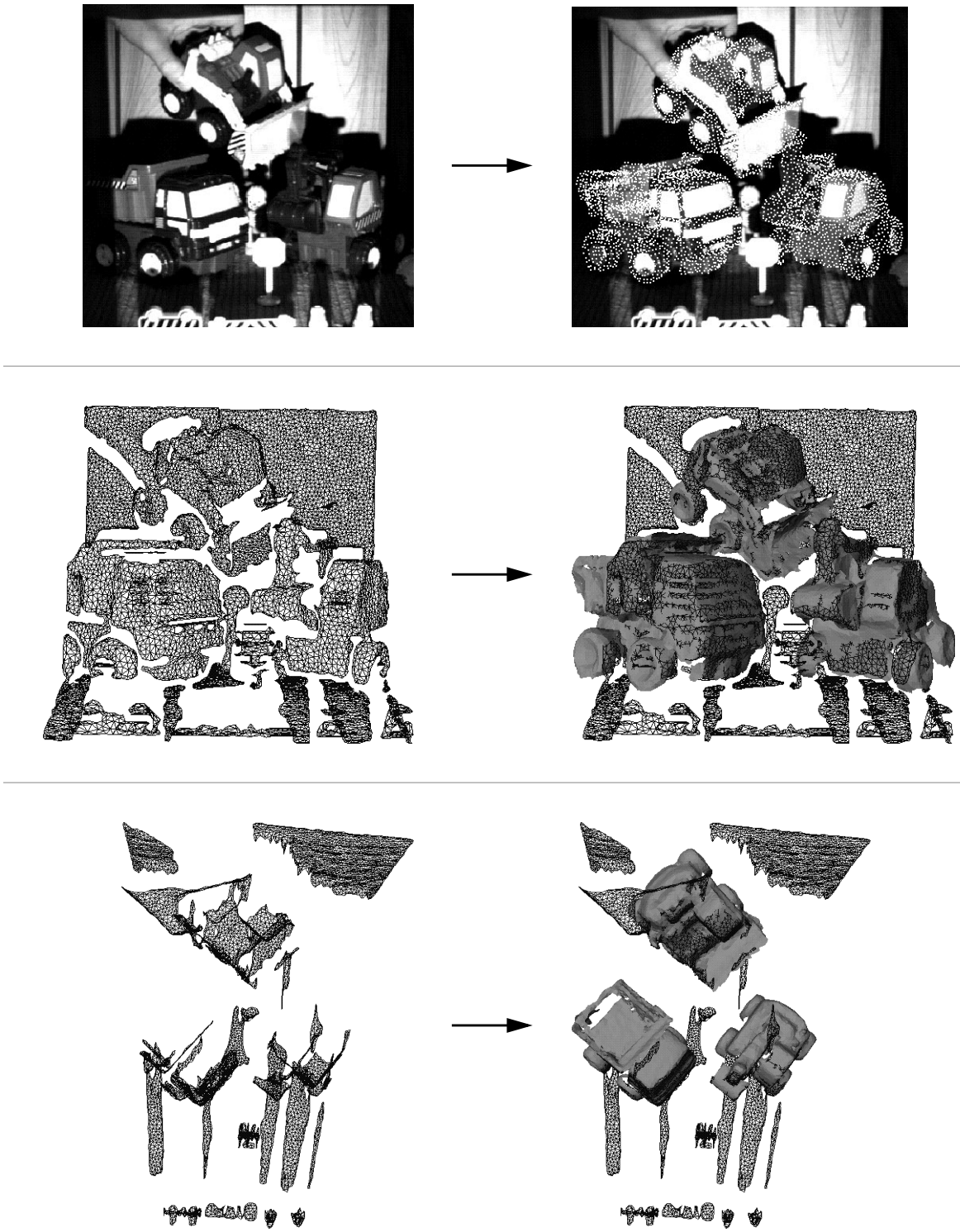


Figure 7-11: Toy Library recognition of 3 similar shaped models using library compression (MA4). Locations of recognized models are shown as model vertices superimposed on the scene image (top), a frontal view of the scene mesh and shaded models (middle), and a top view of scene mesh and shaded models (bottom).

7.2 Quantitative Evaluation of Clutter and Occlusion

Realistic scenes contain clutter and occlusion; any recognition algorithm designed for the real world must work in the presence of clutter and occlusion. We claim that our recognition algorithm is robust to clutter and occlusion and, in Chapter 5, show theoretical models that justify this statement. However, these theoretical models are irrelevant if they are not supported by experimental verification. Therefore, we have developed an experiment to test the effectiveness of our algorithm in the presence of clutter and occlusion. Stated succinctly, the experiment consists of acquiring many scene data sets, running the recognition on the scenes, and then interactively measuring the clutter and occlusion in the scene along with the recognition success or failure. By plotting recognition success or failure against the amount of clutter or occlusion in the scene, the effect of clutter and occlusion on recognition can be determined.

Recognition success or failure can be broken down into four possible recognition states. If the model exists in the scene and is recognized by the algorithm, this is termed a *true-positive* state. If the model does not exist in the scene, and the recognition algorithm concludes that the model does exist in the scene or places the model in an entirely incorrect position in the scene, this is termed a *false-positive* state. If the recognition algorithm concludes that the model does not exist in the scene when it actually does exist in the scene, this is termed a *false-negative* state. Finally, if the model does not exist in the scene and the recognition algorithm concludes that the model does not exist in the scene, then this is termed a *true-negative*. True-positive and true-negative states are recognition successes, while false-negative and false-positive states are recognition failures. However, when using recognition algorithms for some purpose like manipulating objects or navigating a robot, false-positives states are considered worse than false negative states. False positive states deliver false information about the scene, while false-negative states deliver no information since false-negatives are generally interpreted to mean that the object has not been found, rather than that the object does not exist.

In our experiment for measuring the effect of clutter and occlusion on recognition, a *recognition run* consists of the following steps. First, a model is chosen and is placed in the scene with some other objects. The other objects might occlude the object and will produce scene clutter. Next, the scene is imaged and the scene data is processed as described in Section 7.1. A recognition algorithm that matches the model to the scene data is applied, and the result of the algo-

rithm is presented to the user. Using a 3-D interface, the user then interactively segments the surface patch that belongs to the model from the rest of the surface data in the scene. As explained below, once this *model surface patch* has been defined, the amount of clutter surrounding the model surface patch and occlusion of the model are automatically calculated. If the algorithm concludes that the model exists in the scene, it is shown superimposed on the scene surface patch to which it is matched. If the recognition algorithm concludes that the model is not in the scene, it is not shown. Using the presence and location of the model in the scene, the user decides the recognition state, and it is recorded with the amount of clutter and occlusion measured in the scene. By executing many recognition runs using different models and many different scenes, a distribution of recognition state versus the amount of clutter and occlusion in the scene can be generated.

The occlusion of a model is defined as

$$\text{occlusion} = 1 - \frac{\text{model surface patch area}}{\text{total model surface area}} \tag{7.1}$$

For example, if the model surface patch has an area of 40 cm^2 and the model has a surface area of 160 cm^2, the model is 75% occluded. As explained in Chapter 3, surface area for a mesh is calculated as the sum of the areas of the faces making up the mesh. When the model surface patch is segmented from the scene, the occlusion of the model can be automatically calculated by computing the surface patch area and the model surface area. We chose surface area over number of vertices in the surface meshes because we would like a measure of occlusion that is independent of the sampling of the surfaces.

The clutter in the scene is defined as

$$\text{clutter} = \frac{\text{clutter points in relevant volume}}{\text{total points in relevant volume}} \tag{7.2}$$

Clutter points are vertices in the scene surface mesh that are not on the model surface patch. The relevant volume is defined as the union of the support cylinders for all of the spin-images created from oriented points on the model surface patch. In other words, it is the volume around the model surface patch swept out by spin-images from model surface patch oriented points. If the relevant volume contains points that are not on the model surface patch, then these points

will corrupts scene spin-images and are considered clutter points. As shown in our clutter model, the number of points accumulated in a spin-image that are not part of the model surface is the controlling factor in the degradation of spin-image matching using correlation coefficient. Therefore, measuring clutter as the ratio of the number of clutter points in the relevant volume to the total number of points in the relevant volume is a concise description of the amount of model data to non-model data for the spin-images of the model surface patch.

First, we tested the effect of clutter and occlusion on recognition rate using the spin-image without compression (MA1). We selected four models from our library of models: bunny, faucet, Mr. Potato Head and y-split. We then created 100 scenes using these four models; each scene contained all four models. It should be noted that, since all of the scenes contain all four models, a true negative recognition state is not possible. The models were placed in the scene without any systematic method. It was our hope that random placement would result in a uniform sampling of all possible scenes containing the four objects. Examples of eight scenes and recognition runs are shown in Figure 7-21 to Figure 7-28.

For each model, we ran MA1 on each of the 100 scenes, resulting in 400 recognition runs. We limited the libraries to a single object in order to remove the effect of multiple objects from the experiment. Each recognition run was interactively segmented, based on the model being searched for, and the amount of clutter and occlusion was stored with the recognition state. A plot of recognition state in terms of amount of clutter and occlusion for MA1 is shown at the top of Figure 7-12. Each data point in the plot corresponds to a single recognition run; the coordinates give the amount of clutter and occlusion and the symbol describes the recognition state. Briefly looking at the plot shows that the number of true-positive states is much larger than the number of false negative states and false-positive states for matching without compression. Furthermore, as the line in the scatter plot indicates, no recognition errors occur below a fixed level of occlusion, independent of the amount of clutter.

This same procedure using the same 100 scenes was repeated for the matching spin-images with compression (MA2, model dimension $s = 20$) resulting in 400 different recognition runs. Since the model library consists of a single model, testing the algorithm using library compression (MA3 & MA4) was necessary. A plot of recognition state in terms of amount of clutter and occlusion for MA2 is shown at the bottom of Figure 7-12. A brief examination of the plot

shows that the number of true-positive states is much larger than the number of false negative states and false-positive states for matching with compression. As in the case of no compression, no recognition errors occur below a fixed level of occlusion, independent of the amount of clutter.

7.2.1 Effect of Occlusion

Examining the experiment scatter plots in Figure 7-12, one notices that recognition rate is affected by occlusion. At low occlusion values, no recognition failures are reported, while at high occlusion values, recognition failures dominate. This indicates that recognition will almost always work if sufficient model surface area is visible. The decrease in recognition success after a fixed level of occlusion is reached (70%) indicates that spin-image matching does not work well when only a small portion of the model is visible. This is no surprise since spin-image descriptiveness comes from accumulation of surface area around a point. In these recognition runs, the surface area threshold for reporting matches is set at 0.1. Therefore, surface matches with matched model surface area less than roughly 10% (90% occlusion) will not be reported, making positive recognitions for occlusion greater than 90% very rare.

Successful recognition rate is high below a fixed level of occlusion.
Above this level, recognition success decreases as occlusion increases.

To quantify the effect of occlusion, we generated cumulative distribution functions (CDF) for each possible recognition state versus percent occlusion. Figure 7-13 shows these plots for matching with and without compression. A data point *(occlusion, recognition rate) = (O,R)* on an occlusion CDF has the following interpretation. If the occlusion in the scene is less than *O,* then the recognition rate is *R*. Recognition rate is the number of recognition runs of a particular state (e.g., true-positive) with occlusion less than *O* divided by the total number of recognition runs with an occlusion less than *O*.

The plots show that true-positive recognitions fall off as the occlusion increases, and consequently, false positive and false negative rates increase. However, the success of recognition still remains high, no recognition failures are reported up to around 70% occlusion. After this, successful recognitions gradually decrease until occlusion reaches 90%; recognition ceases after 90% because the user-supplied surface area threshold prevents matches with small surface area from being reported.

The occlusion CDF plots in Figure 7-13 show the effect of occlusion on recognition independent of the amount of clutter. To show the effect of clutter on recognition while occlusion is varying, we have created the plots shown in Figure 7-14 and Figure 7-15. These plots should be interpreted as follows. Each occlusion CDF is computed from the set of recognition runs whose clutter is less than the specified level. For example, the occlusion CDF for a clutter level of less than 40% includes only the recognition runs whose clutter measurement is less than 40%. Showing the occlusion CDFs for different clutter levels shows the effect of occlusion on recognition state for increasing amounts of clutter. Since the recognition rates are almost the same for different clutter levels, recognition rates are basically unaffected by clutter.

Probability distribution functions (PDFs) are easier to understand than are CDFs. Constructing PDFs requires a decision on how large the bins for calculating probabilities will be. Bins must be large enough to enable calculation of a meaningful probability, but small enough to represent the probability for small interval of occlusion. With the relatively small number of experiments, deciding on an appropriate bin size for PDFs describing recognition rate versus occlusion is difficult.

We generate PDFs for recognition rate versus occlusion as follows. Instead of discretely binning the recognition runs to compute recognition rate, the recognition rate at a particular level of occlusion are computed using a gaussian weighted average. Suppose that we want to calculate the recognition rate a level of occlusion V. For a recognition run with a level of occlusion O, a weight W is computed using

$$W = \exp\left(-\left(\frac{(O - V)}{\sigma_V}\right)^2\right) \tag{7.3}$$

The true positive recognition rate R_V at level of occlusion V is the sum of the weights from recognition runs with true positive state divided by the total weights from all recognition runs. Similar definitions exist for false negative and false positive states. To compute the occlusion PDF, recognition rates are calculated for equally spaced levels of occlusion V, and σ_V in (7.3) is set to the spacing between occlusion levels. By using the gaussian weighted average to compute recognition rate, some of the binning problems associated with creating a PDF are avoided.

The PDFs shown in Figure 7-16 validate our claim about occlusion. Recognition rate remains high for both forms of compression until occlusion of around 70% is reached; then the successful recognition rate begins to fall off.

7.2.2 Effect of Clutter

Examining the experiment scatter plots in Figure 7-12, one notices that the effect of clutter on recognition is uniform across all levels of occlusion. At low occlusion values no recognition failures are reported, while at high occlusion values, recognition failures dominate, independent of the amount of clutter in the scene This indicates that spin-image matching is independent of the clutter in the scene.

Recognition is resistant to clutter.

To quantify the effect of clutter, we generated cumulative distribution functions versus percent clutter for each possible recognition state. Figure 7-17 shows these plots for matching with and without compression. A data point *(clutter, recognition rate) = (C,R)* on a clutter CDF has the following interpretation. If the clutter in the scene is less than *C*, then the recognition rate is *R*. Recognition rate is the number of recognition runs of a particular state (e.g., true-positive) with clutter less than *C* divided by the total number of recognition runs with a clutter level less than *C*.

The initial increase in true-positive rate as clutter increases from zero is due to the large number of recognition runs with low clutter and high occlusion. This forces the true-positive clutter CDF to start off low but then increase. After a clutter level of 10%, the recognition rates level off. Once again, this indicates that recognition success is independent of clutter level.

The clutter CDF plots in Figure 7-17 show the effect of clutter on recognition independent of the amount of occlusion. To show the effect of clutter on recognition while occlusion is varying, we have created the plots shown in Figure 7-18 and Figure 7-19. These plots should be interpreted as follows. Each clutter CDF is computed from the set of recognition runs whose occlusion is less than the specified level. For example, the clutter CDF for a occlusion level of less than 70% includes only the recognition runs whose occlusion measurement is less than 70%. Showing the clutter CDFs for different occlusion levels shows the effect of clutter on recognition state for increasing amount of occlusion. The recognition rates are all basically flat for

different occlusion levels. However, as occlusion increases, true-positive recognition decreases. This is consistent with our conclusion that occlusion controls recognition rate.

As with occlusion, we have included clutter PDFs for clarity. The plots in Figure 7-20 show the recognition rate PDFs as clutter increases. The PDFs show that recognition rate is fairly independent of clutter. As clutter increases, the clutter PDFs have slight variations about a fixed recognition rate. Most likely, these variations are due to non-uniform sampling recognition runs and are not actual trends with respect to clutter. Above a high level of clutter, the successful recognitions decline. However, from the scatter plots, we see that at high levels of clutter, the number of experiments is small, so conclusions about recognition rate should not be made.

7.2.3 Effect of Spin-Image Compression

In all of the plots showing the effect of clutter and occlusion, the true-positive rates are higher for recognition with spin-images without compression when compared to the true-positive rates for recognition with compression. This validates the expected decrease in the accuracy of spin-image matching when using compressed spin-images. However, it should be noted that the true-positive recognition rates for both matching algorithms remain high. Matching without compression has an average recognition rate for all recognition runs of 90.0% and matching with compression has an average recognition rate of 83.2%. Furthermore, the false-positive rate for both algorithms are low and nearly the same; for MA1 it is 3.5% and for MA2 it is 1.8%. The difference in true-positive rates comes from the difference in false negative rates; 6.5% for MA1 and 15.0% for MA2. Although matching with spin-image compression is worse, the degradation in performance comes from an increase in false negative recognitions, which are not as detrimental to a robotic system as false positive recognitions. As shown in the next section, the decrease in recognition rate for matching with compression is made up for by a large increase in matching speed.

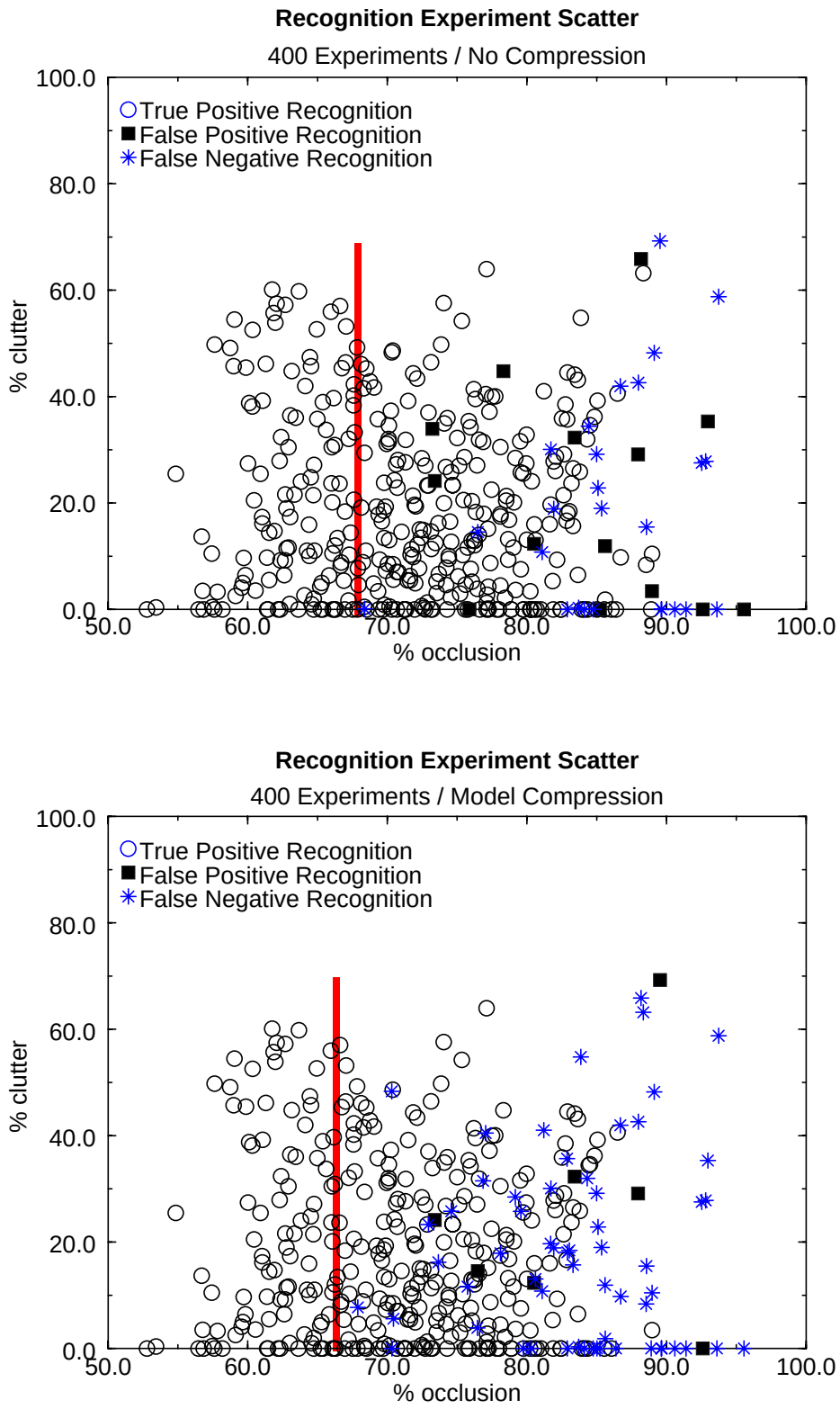


Figure 7-12: Plots of recognition state in terms of clutter and occlusion for 400 recognition runs. Recognition without compression (top) achieves more true-positive recognitions than recognition with compression (bottom). The line in each plot indicates the occlusion level below which no recognition failures are reported.

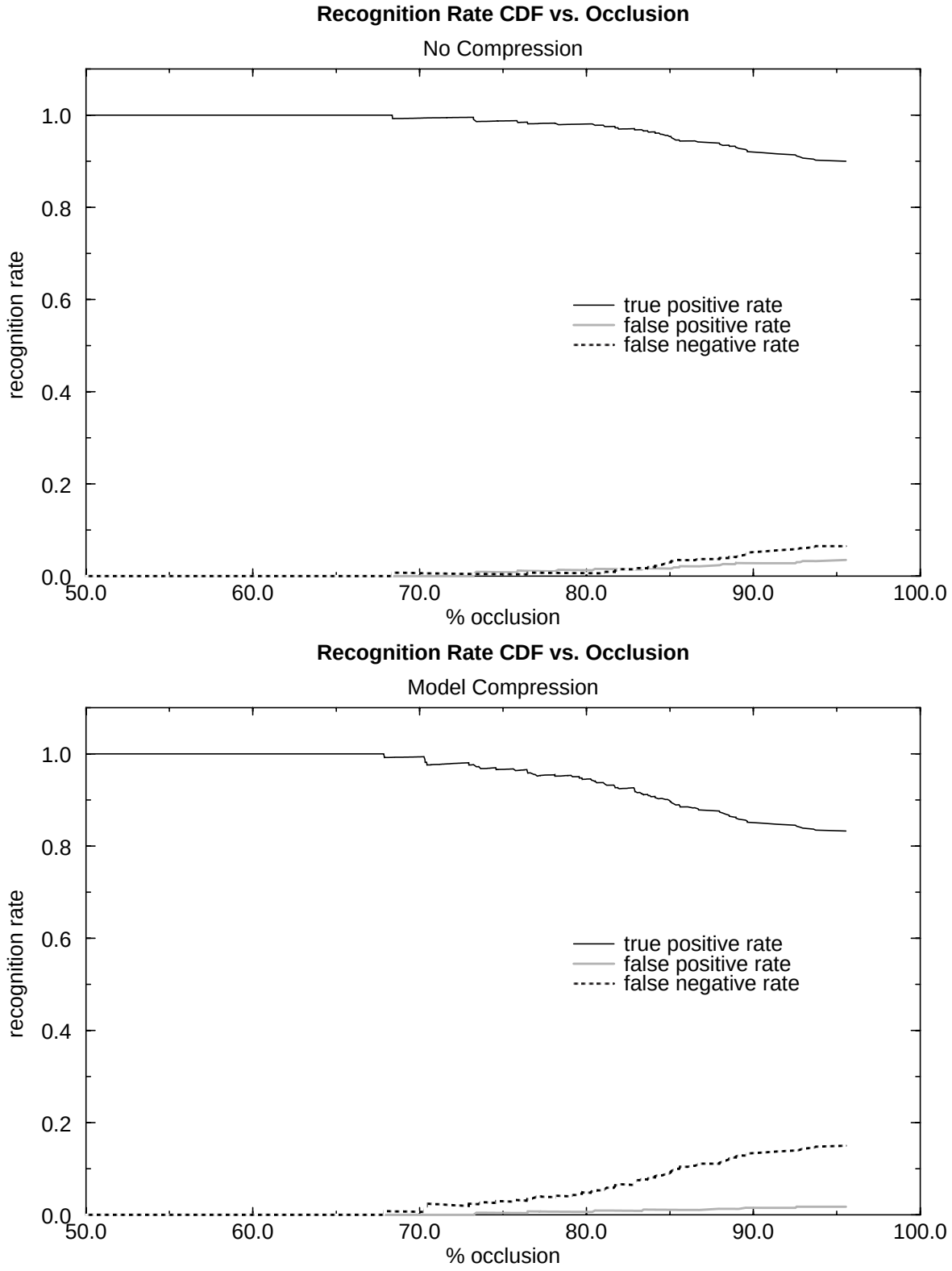


Figure 7-13: Plots of recognition rate versus amount of occlusion in scenes. Recognition without compression (top) has a higher true-positive rate than recognition without compression (bottom). Recognition success decreases as occlusion increases.

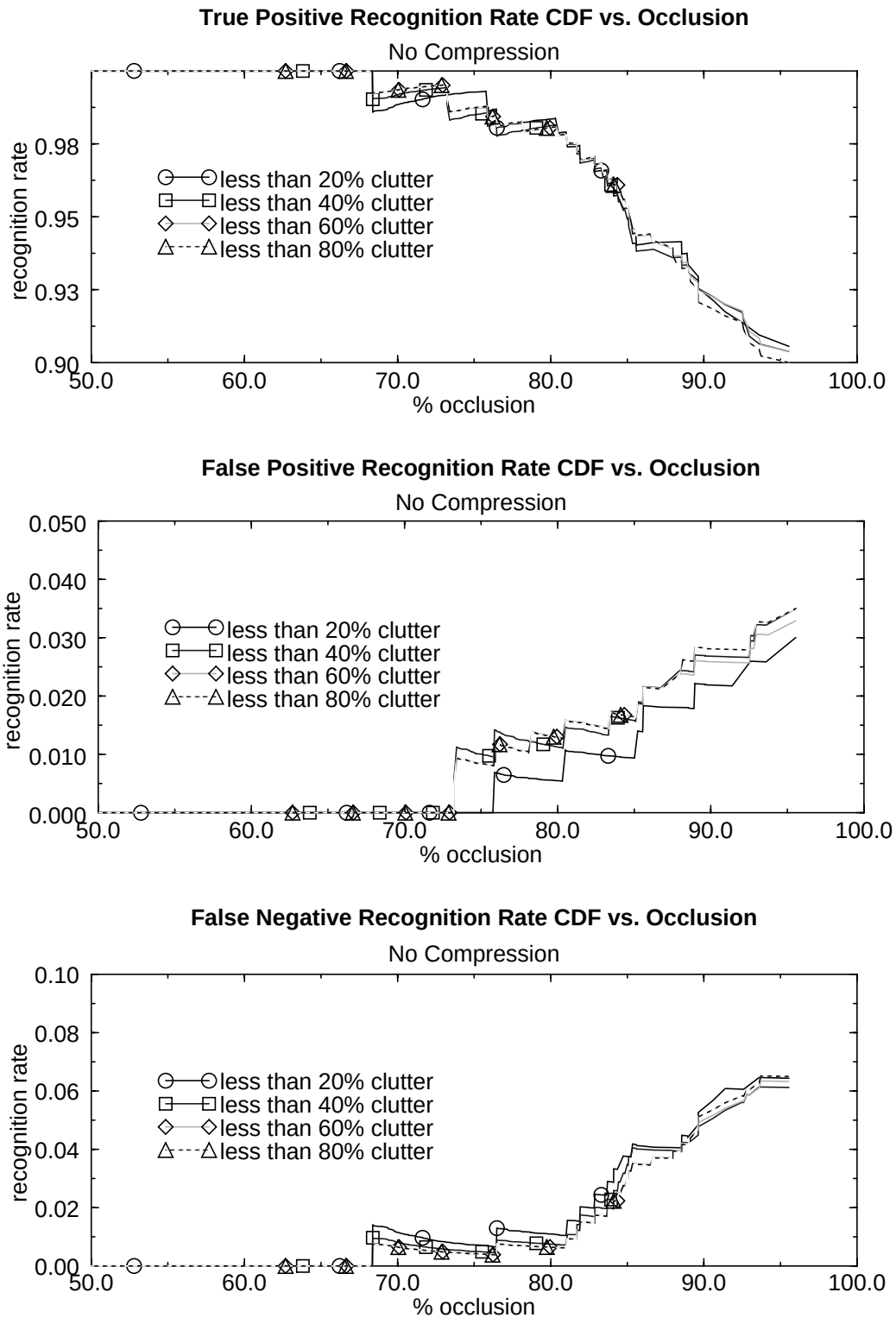


Figure 7-14: Plots of recognition rates versus amount of occlusion in scenes for varying levels of clutter for matching without compression. The recognition rates are essentially the same independent of the amount of clutter in the scene.

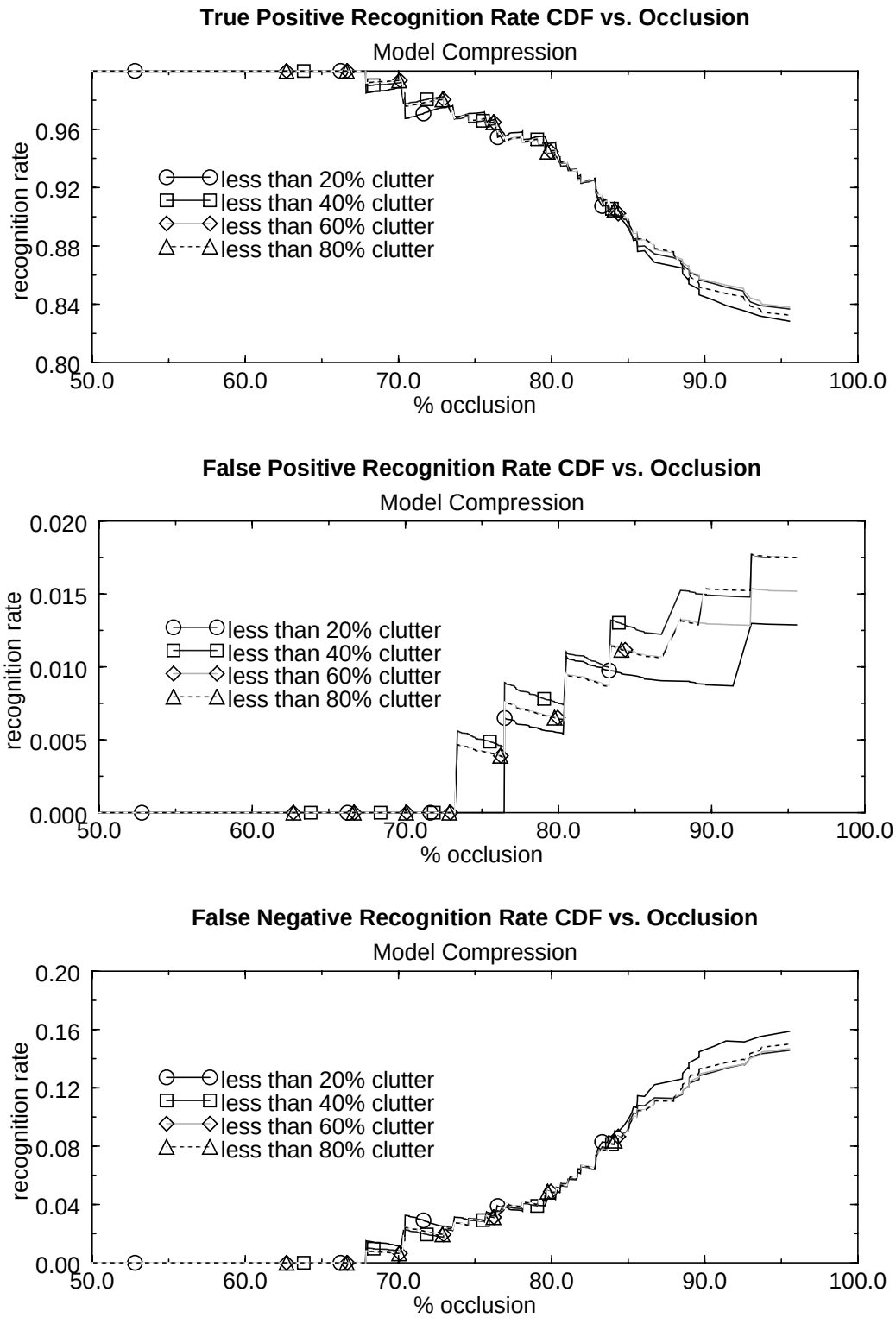


Figure 7-15: Plots of recognition rates versus amount of occlusion in scenes for varying levels of clutter for matching with compression. The recognition rates are essentially the same independent of the amount of clutter in the scene.

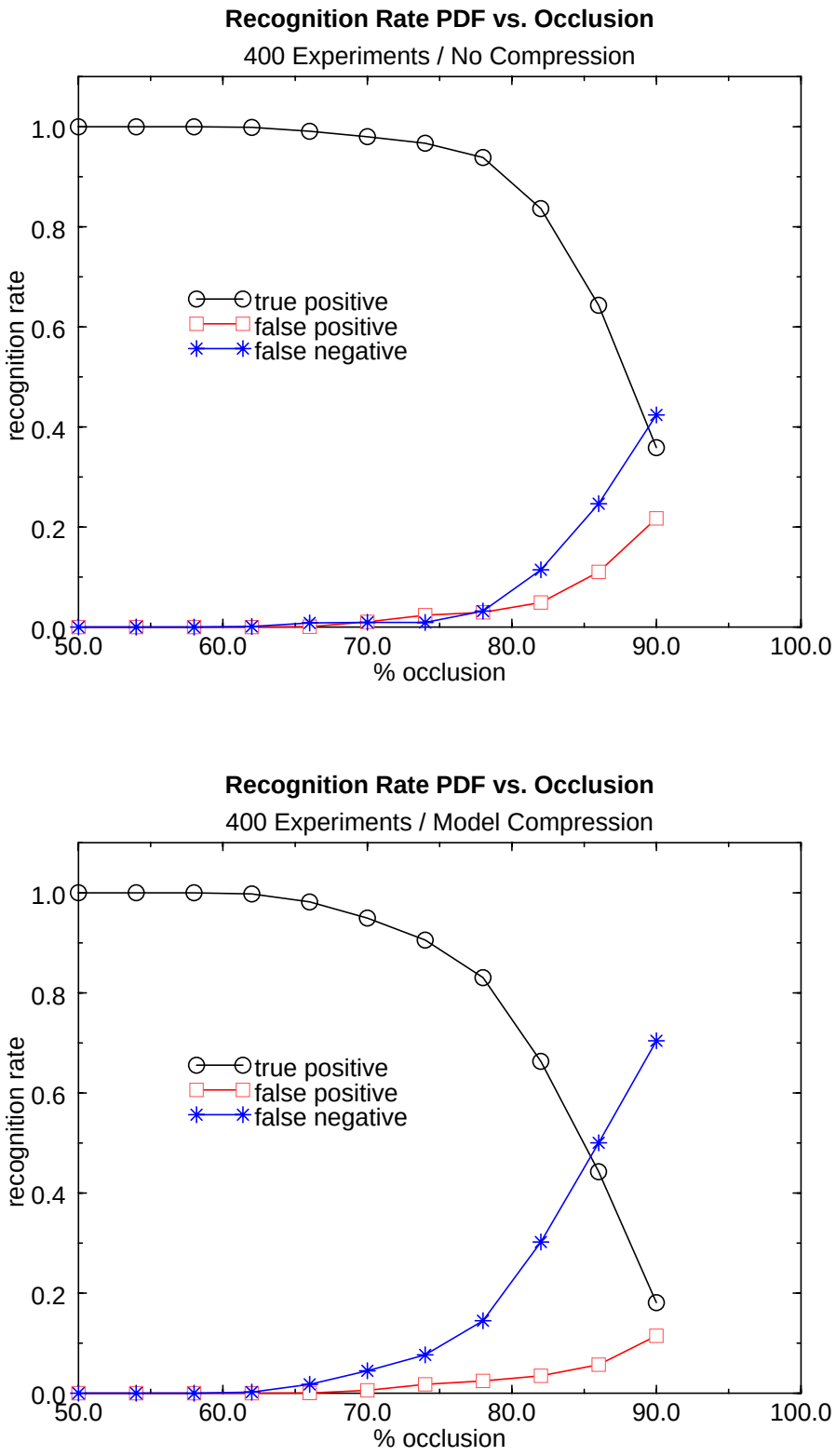


Figure 7-16: Recognition rate PDFs versus occlusion for spin-image matching with and without compression. As occlusion increases, successful recognitions decrease, but the number of false negative states remains higher than the number of false positive states.

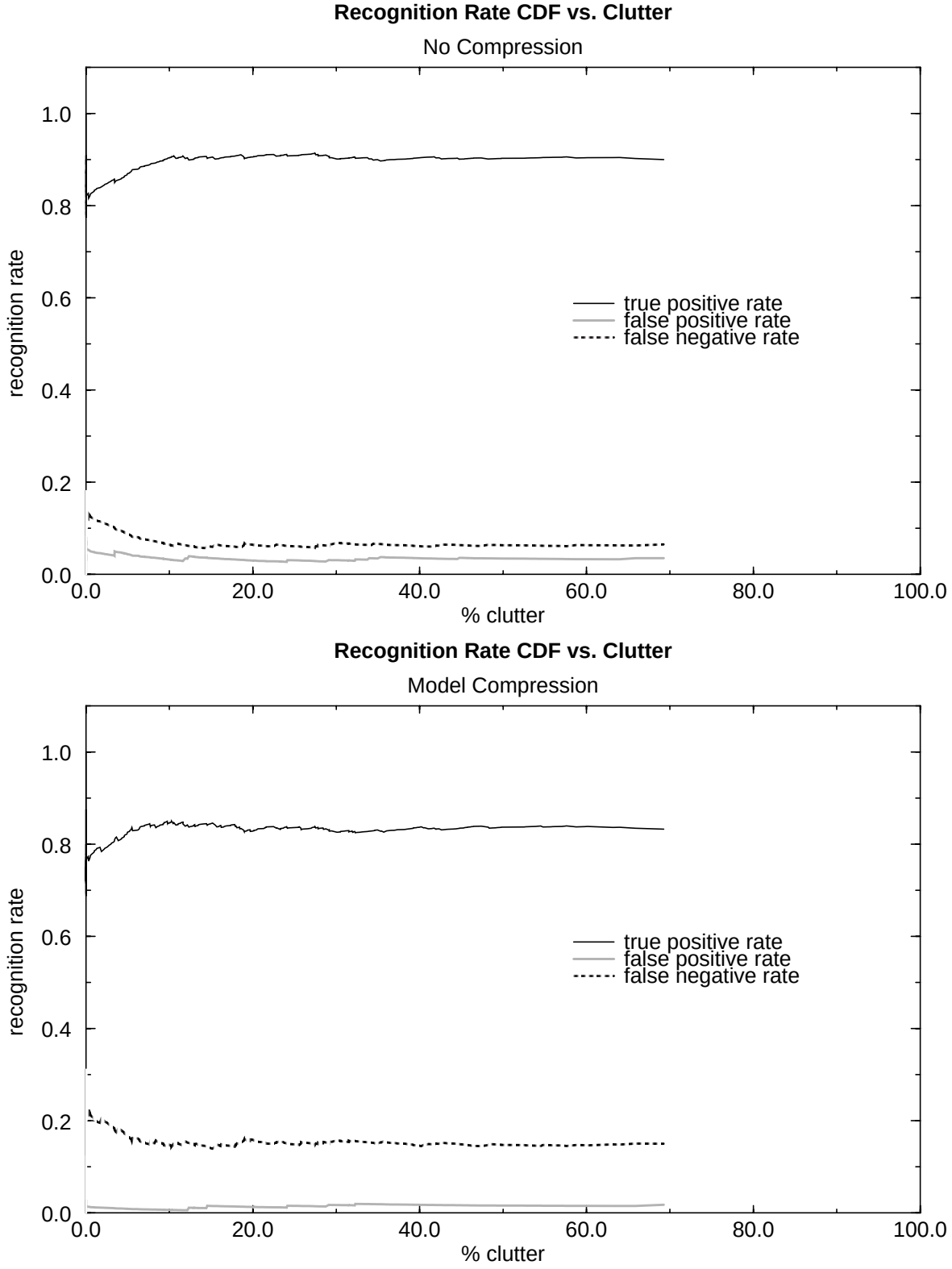


Figure 7-17: Plots of recognition rate versus amount of clutter in scenes. Recognition without compression (top) has a higher true-positive rate than recognition without compression (bottom). Both show that clutter has little effect on recognition.

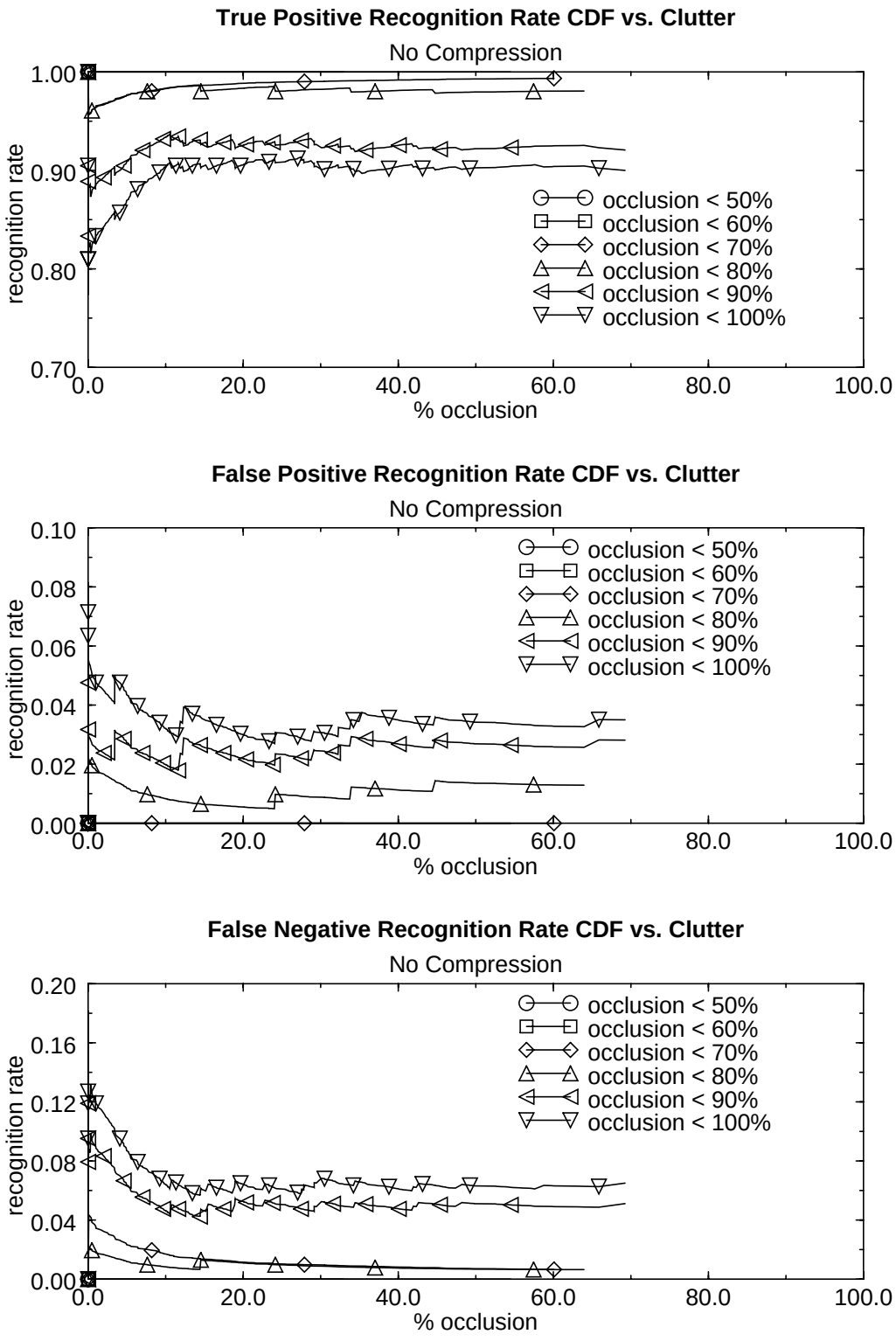


Figure 7-18: Plots of recognition rates versus amount of clutter in scenes for varying levels of occlusion for matching without compression. Rates change as occlusion level changes, but remain fixed as clutter increases. This validates our claim that recognition is independent of clutter, but depends on occlusion.

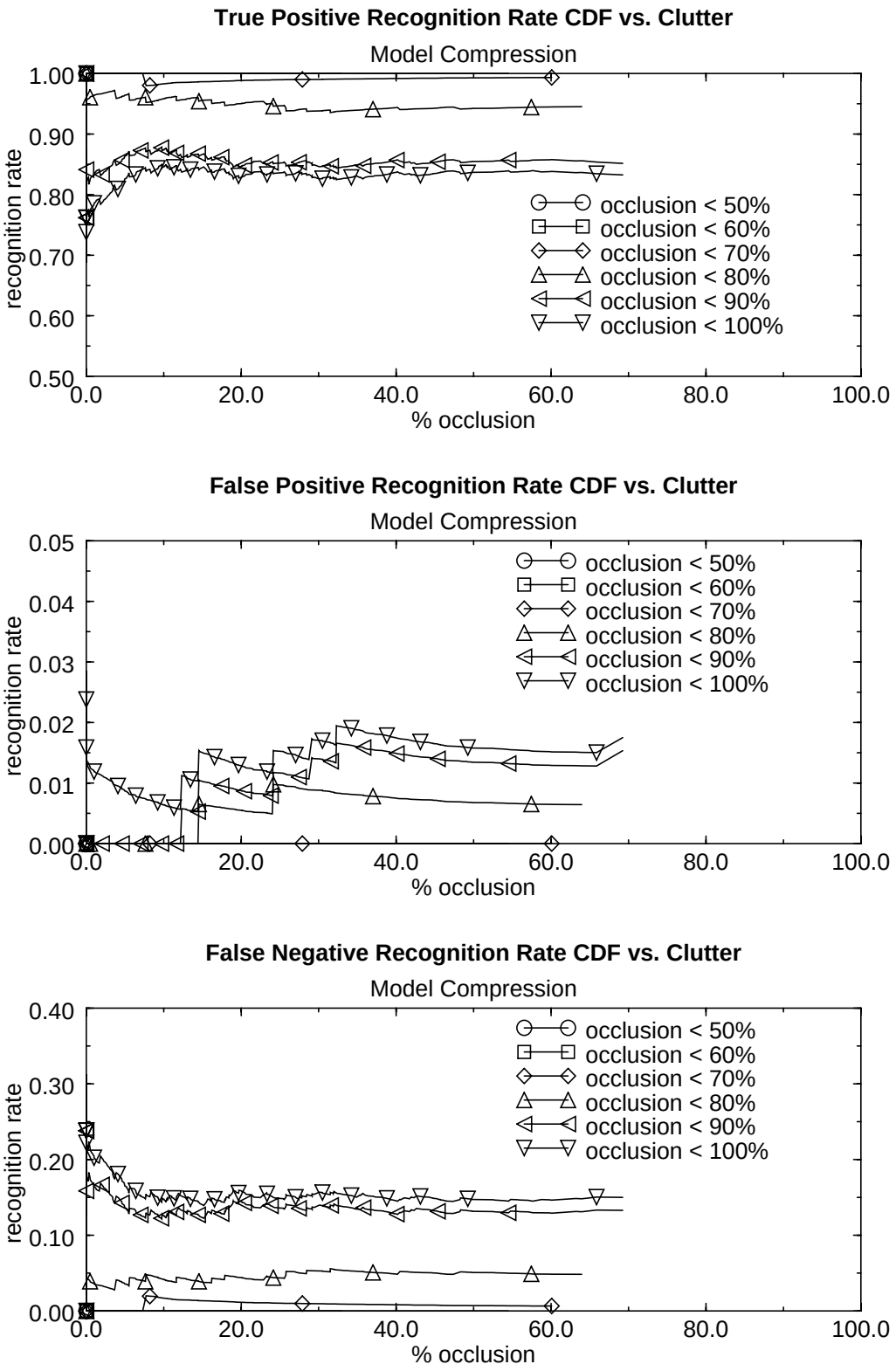


Figure 7-19: Plots of recognition rates versus amount of clutter in scenes for varying levels of occlusion for matching with compression. Rates change as occlusion level changes, but remain fixed as clutter increases. This validates our claim that recognition is independent of clutter, but depends on occlusion.

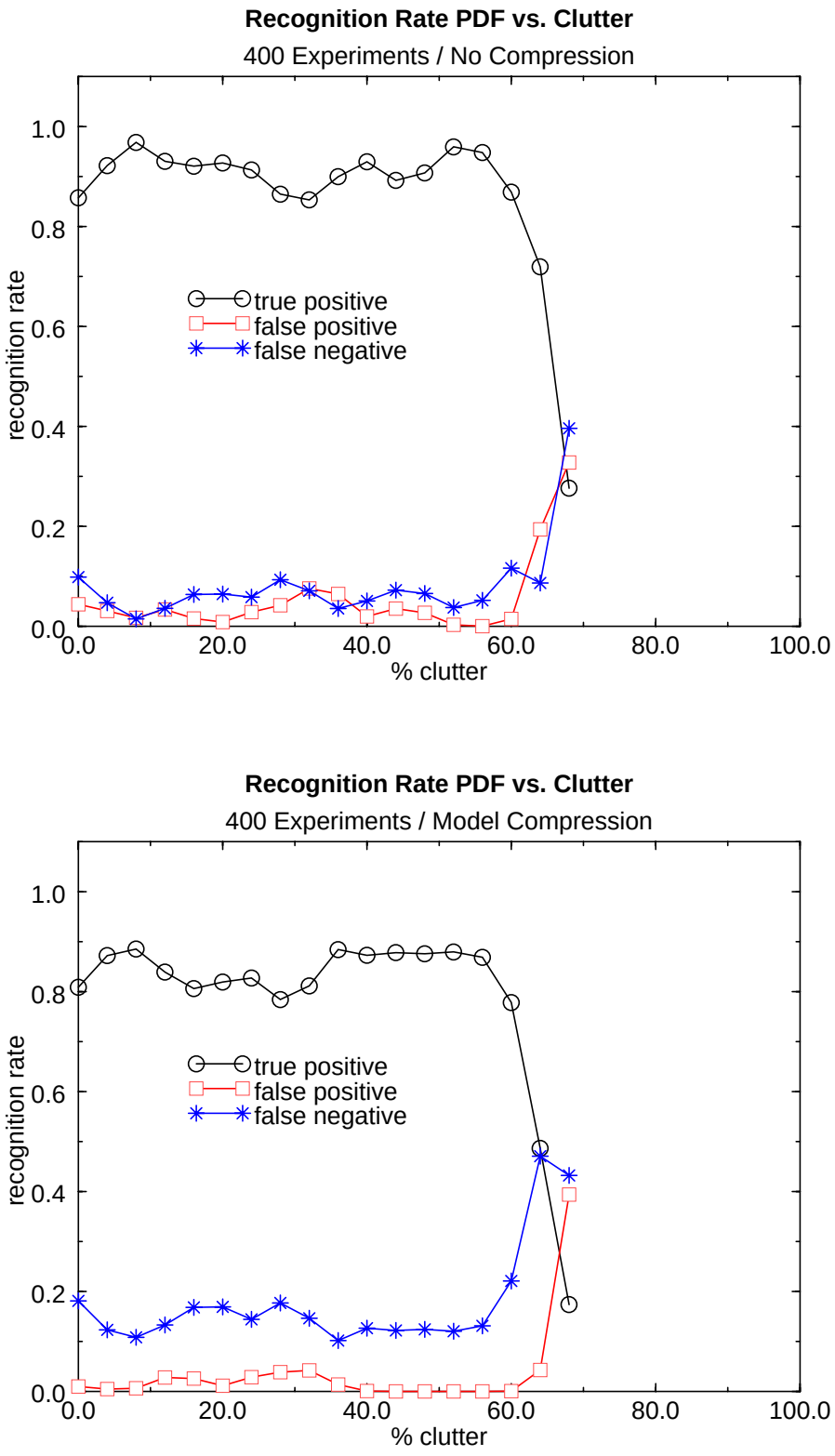


Figure 7-20: Recognition rate PDFs versus clutter for spin-image matching with and without compression. As clutter increases, recognition rates stay fairly independent of clutter. The variations from horizontal in the plots are due to experimental variations, not necessarily to actual trends in recognition rate. When clutter increases above a high level, successful recognition rate drops off.

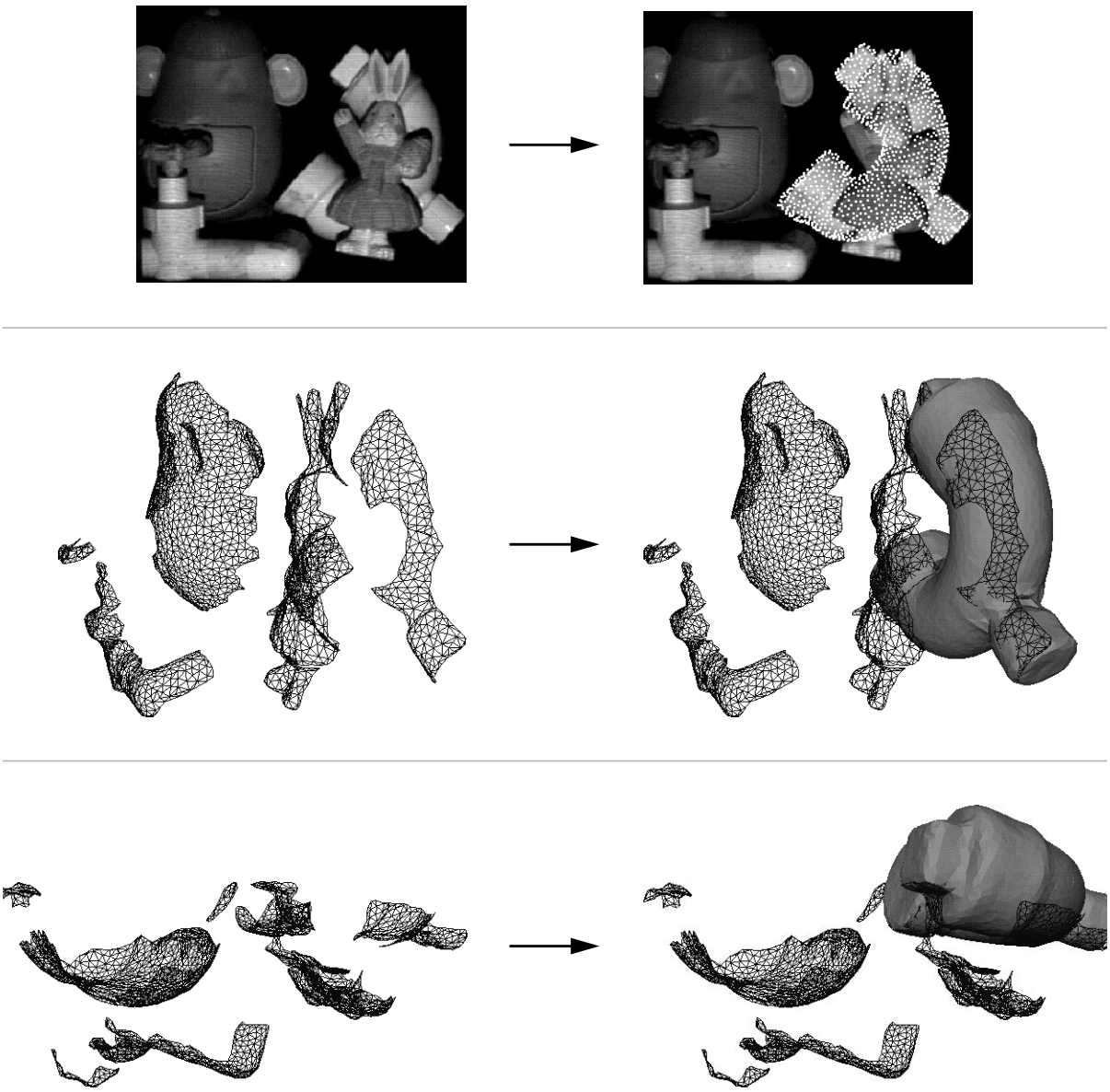


Figure 7-21: Recognition of the y-split model without spin-image compression. The three views of the scene and recognized model show that the scene is cluttered (55.56%) and the model is occluded (84.1%).

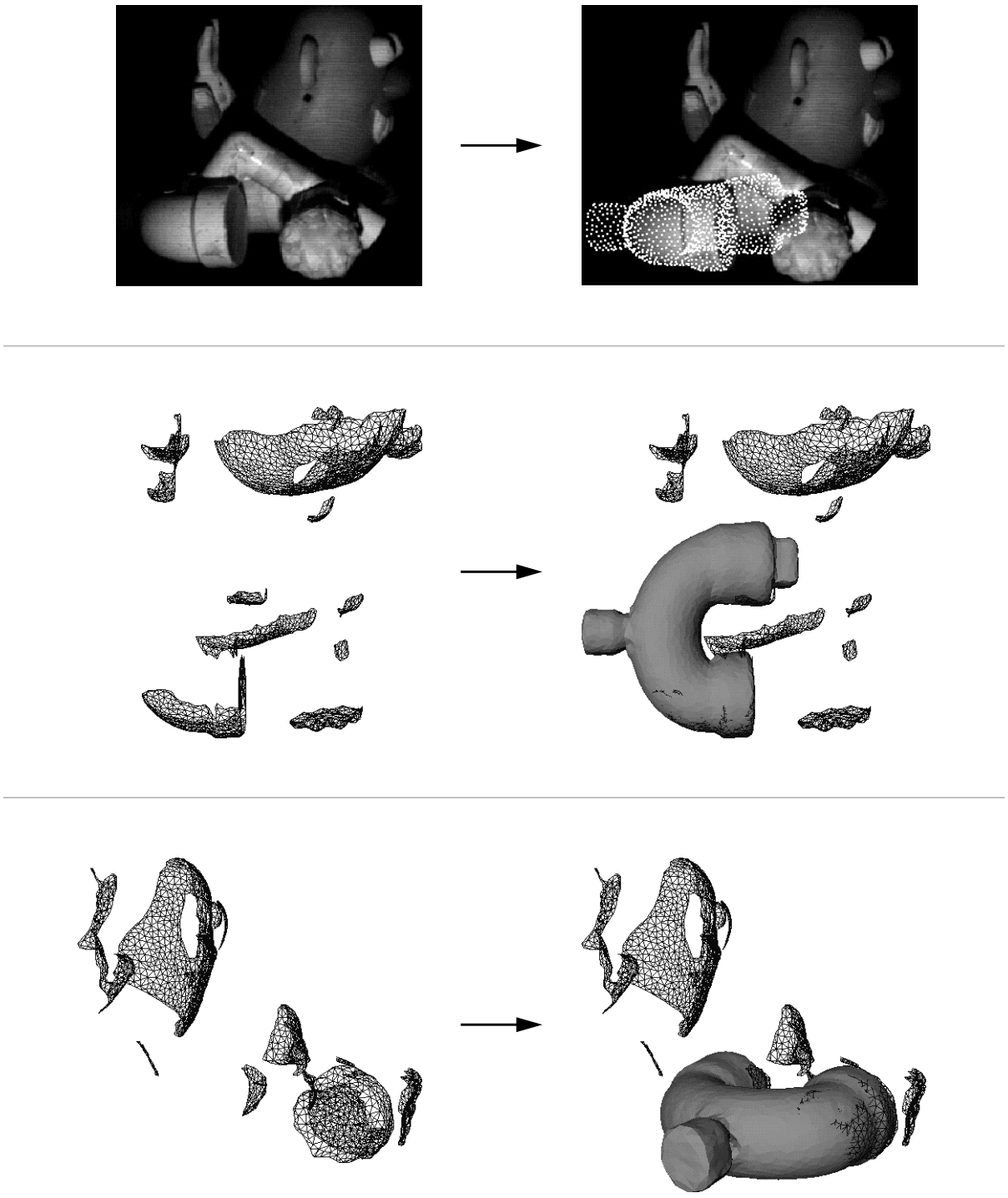


Figure 7-22: Recognition of the y-split model with spin-image compression. The three views of the scene and recognized model show that the scene is cluttered (40.6%) and the model is occluded (86.5%).

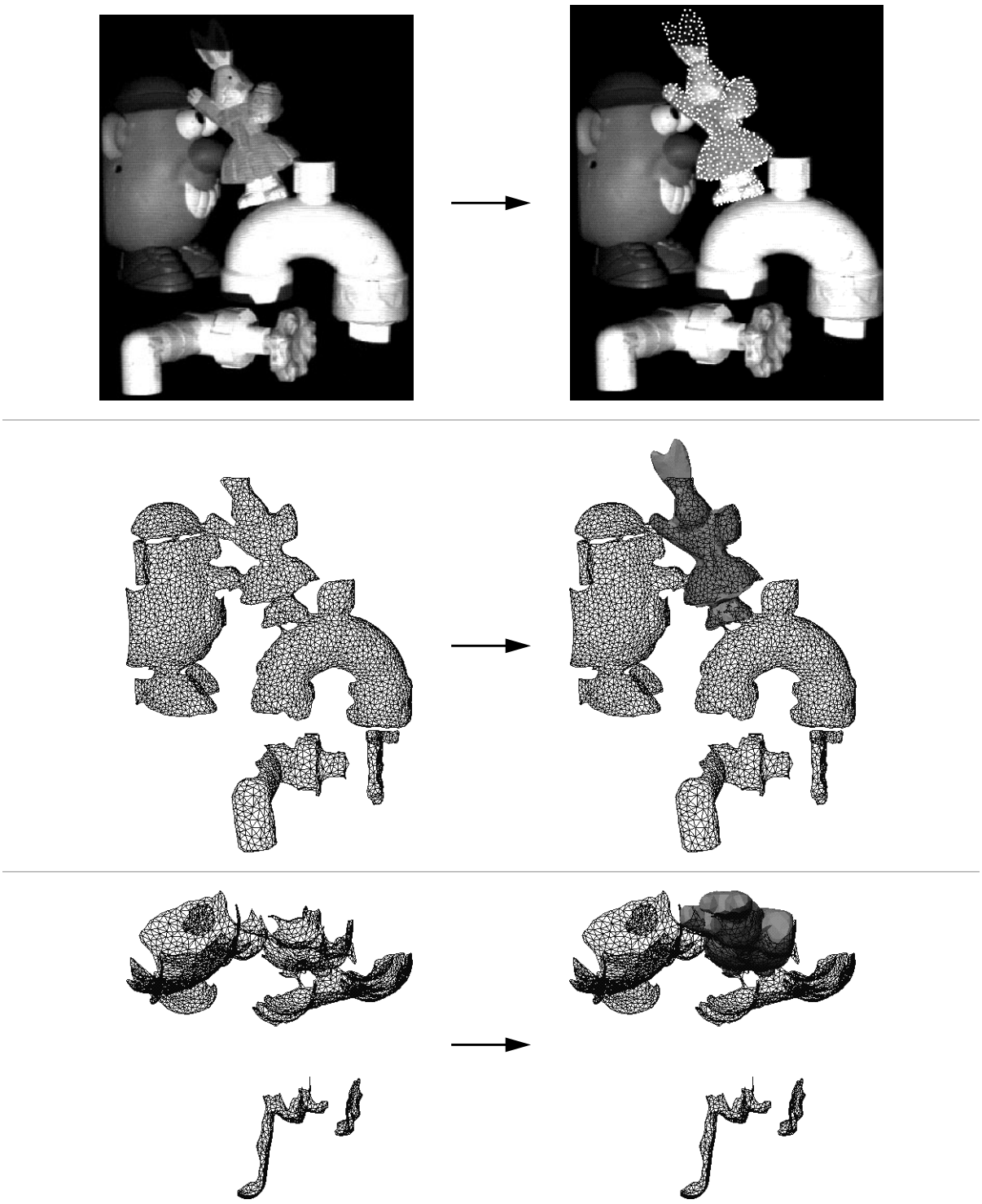


Figure 7-23: Recognition of the bunny model without spin-image compression. The scene has 59% clutter and 63% occlusion.

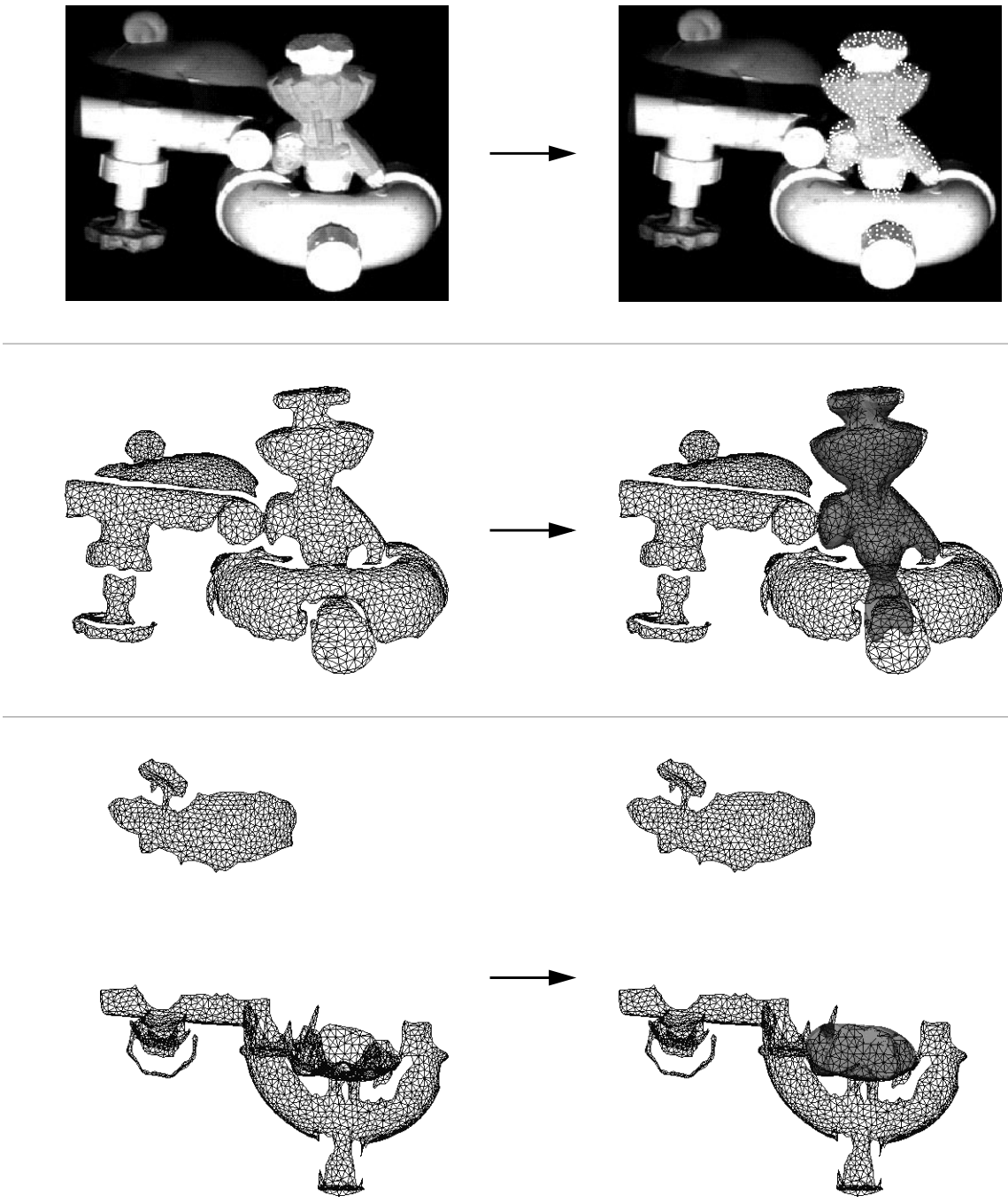


Figure 7-24: Recognition of the bunny model with spin-image compression. The scene has 59% clutter and 65% occlusion.

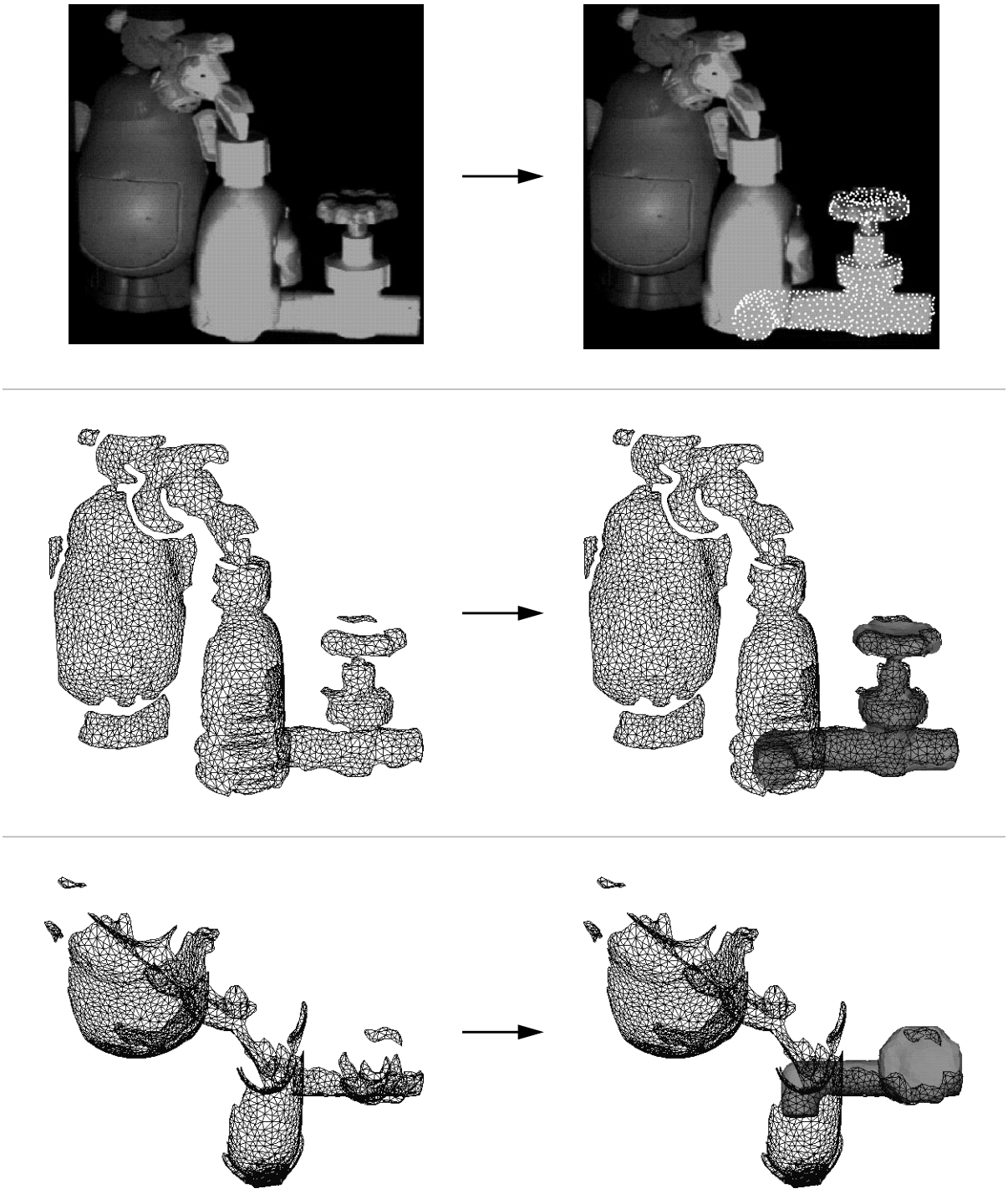


Figure 7-25: Recognition of the faucet model without spin-image compression. The scene has 32% clutter and 70% occlusion.

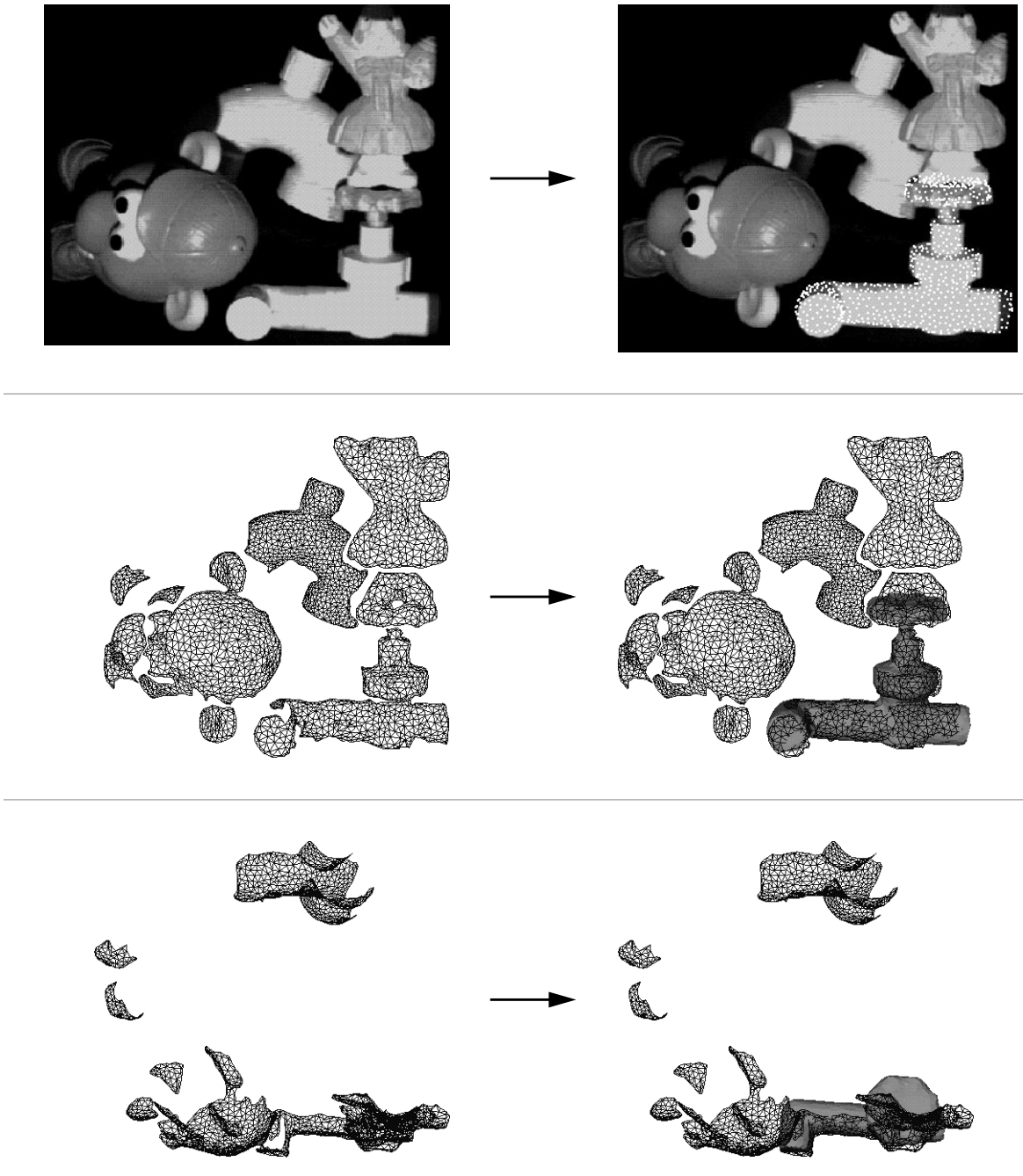


Figure 7-26: Recognition of the bunny model with spin-image compression. The scene has 43% clutter and 68% occlusion.

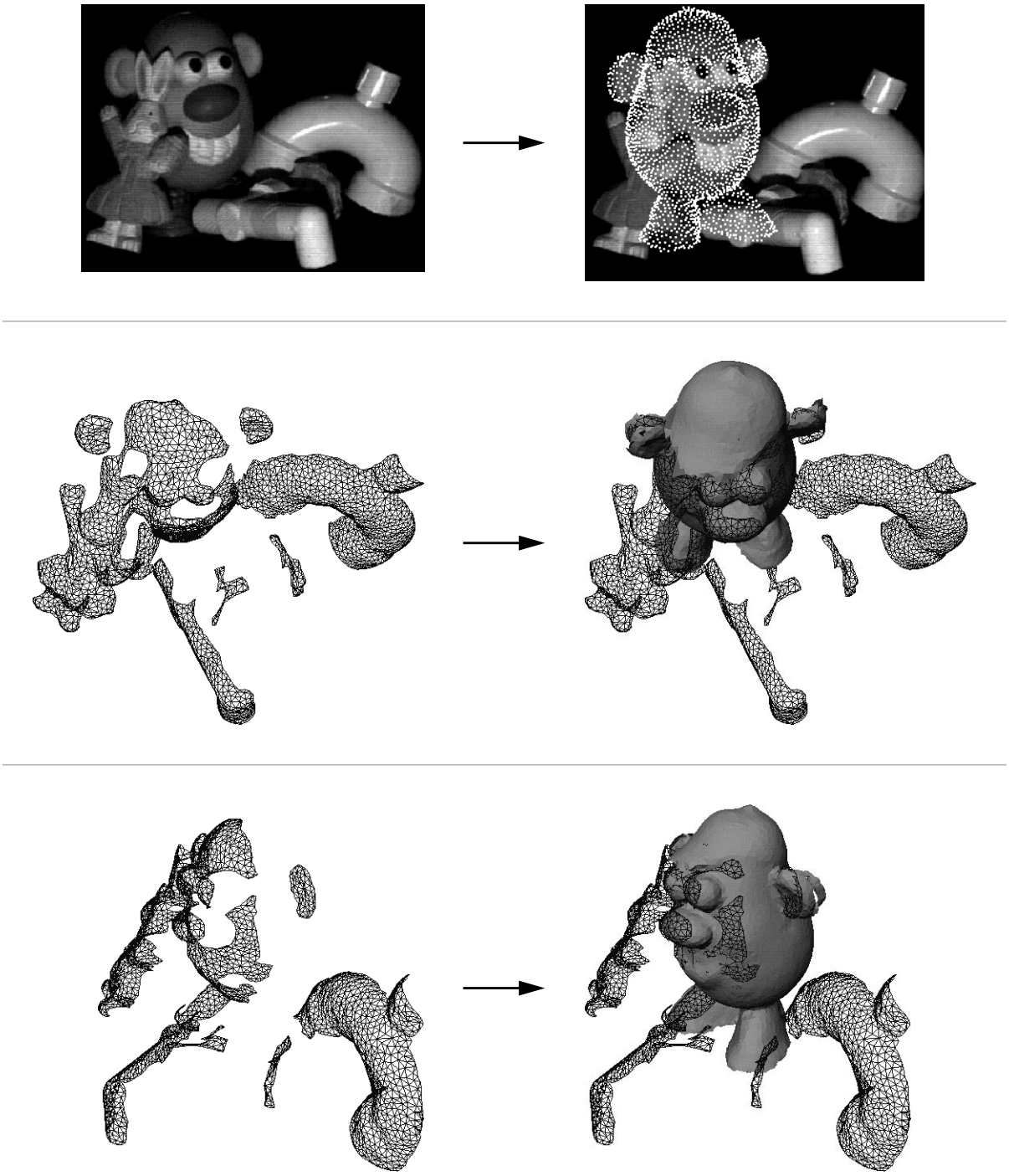


Figure 7-27: Recognition of the Mr. Potato Head model without spin-image compression. The scene has 45% clutter and 84% occlusion.

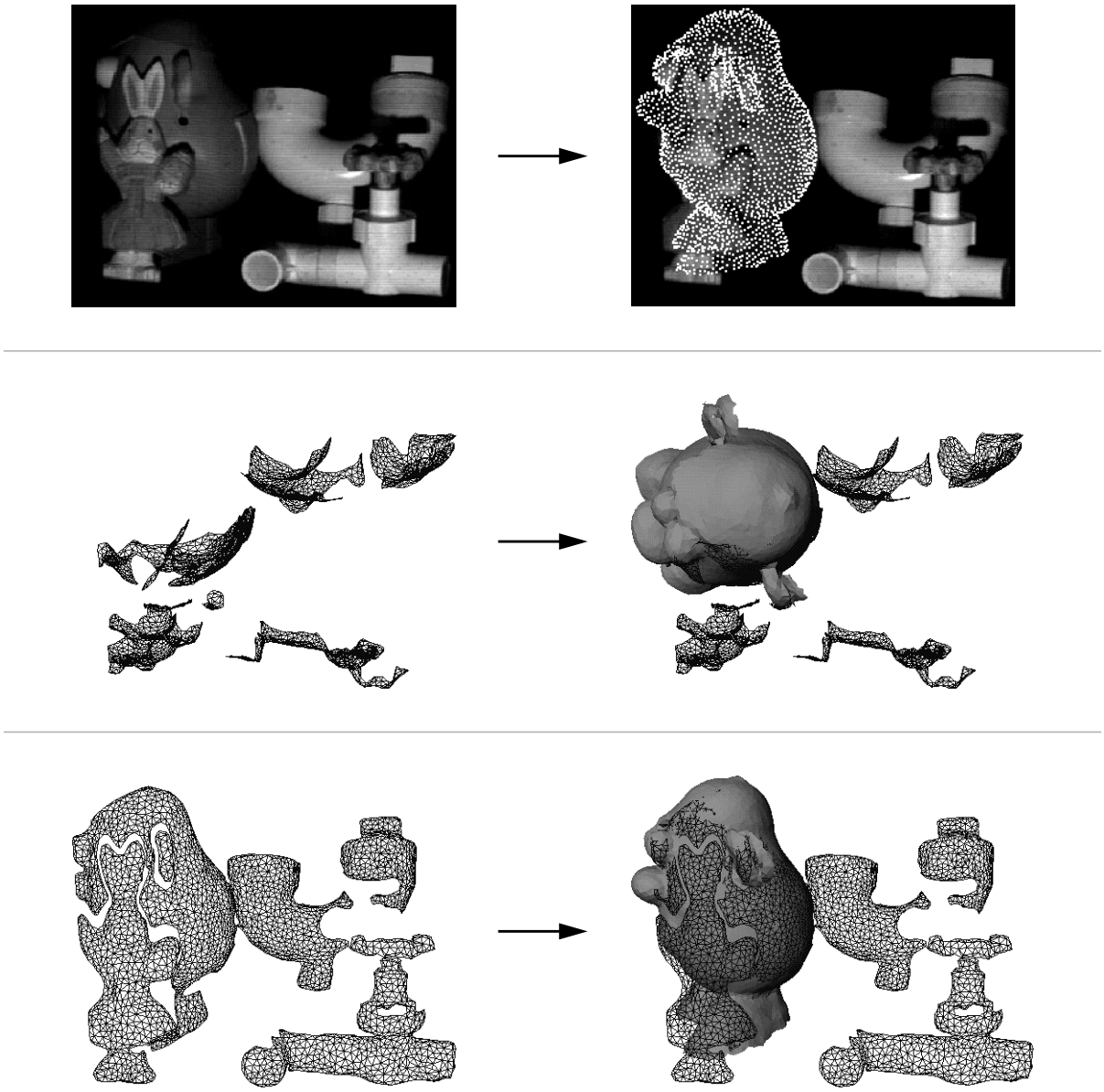


Figure 7-28: Recognition of the Mr. Potato Head model with spin-image compression. The scene has 39% clutter and 83% occlusion.

7.3 Effect of Library Size

Increasing the number of models in a model library will increase the number of objects that can be searched for in the scene. Unfortunately, it can also increase the chance of incorrect matches between spin-images because individual spin-images from one model may be duplicated by spin-images in another model. The purpose of this section is to evaluate the performance of four matching algorithms described in Chapter 6 in terms of recognition rates and running times.

To test the algorithm, the following experiment was performed. Model libraries were made with an increasing number of models from models shown in Figure 7-1; the smallest library contained just the duckie model, while the largest library contained all twenty models. Next, a scene containing various combinations of the twenty objects, but always containing the duckie model, was created. The scene was then matched to each of the model libraries using each of the four matching algorithms. For each recognition run, the running times of the algorithm, the number of models incorrectly recognized (false-positive), the number of models correctly recognized (true-positive) and of these the number of models correctly localized, were recorded. The above process was repeated for ten scenes, four of which are shown in Figure 7-2 through Figure 7-5.

7.3.1 Recognition Rate

The top of Figure 7-29 shows the true-positive recognition rate for the four matching algorithms versus the number of models in the model library. True positive recognition rate is calculated as the number of true-positive recognitions in all ten scenes divided by the number of true-positive and false positive recognitions in all ten scenes. It does not measure the number of false-negative recognitions. The plot shows that the recognition rate is high (about 90%) independent of the number of models in the library. Furthermore, none of the algorithms performs significantly better than the others. Since true-positive recognition rate remains fixed, it has been demonstrated that adding models to the model library does not degrade spin-image matching.

The middle of Figure 7-29 shows the localization rate for the four matching algorithms versus the number of models in the model library. Localization rate is the number of correctly local-

ized and recognized (true-positive) models in all ten scenes divided by the number of true-positive and false positive recognitions in all ten scenes. Localization rate will always be less than true-positive recognition rate. Once again, localization rate remains fixed for all matching algorithms (about 80%), independent of the number of models in the model library.

The bottom of Figure 7-29 shows the average number of true positive recognitions for all ten scenes versus the number of models in the model library. As the number of models increases, the number of models correctly recognized increases linearly. This is caused by models that are present in the scene being added to the model library as the number of models increases. Therefore, more models can be matched to the scene. Matching without compression matches slightly more models than the other algorithms because uncompressed spin-images are more discriminating.

7.3.2 Running Times

The time needed to match a single scene spin-image to all of the spin-images in the model library as the number of models in the library increases is shown in Figure 7-30. All times are real wall clock times on a Silicon Graphics O2 with a 174 MHz R10000 processor. As expected, matching of spin-images grows linearly with the number of models in the model library because the number of spin-images being compared increases linearly with the number of models. This is true of matching with compression and matching without compression; however, the matching times with compression grow significantly slower than the matching times without compression. With twenty models, the MA1 algorithm takes 20 times longer that the MA3 and MA4 algorithms and the 10 times longer than the MA2 algorithm; both are order of magnitude improvements. Since there is only a slight decrease in recognition performance for the algorithms that use compression, while running times are dramatically improved, compressed spin-images should be used in recognition. Another factor in matching time, is the number of points in the scene. To obtain the total match time for the algorithms, the match times shown in Figure 7-30 should be multiplied by the number of points selected from the scene for matching.

Correspondence grouping times for the four algorithms are shown in the top of Figure 7-31. In light of the time it takes to match spin-images, the correspondence grouping times are insignif-

icant. The grouping times for the MA1 algorithm are slightly higher than the other algorithms. This is probably caused by a larger number of correspondences between model and scene when matching without compression because of the higher discrimination between spin-images.

Surface match verification times for the four algorithms are shown at the bottom of Figure 7-31. For all of the algorithms, the verification times grow linearly with the number of models in the model library. This is directly attributable to the increasing number or recognized models as shown at the bottom of Figure 7-29. As the number of models in the model library increases, the number of models that can be matched in the scene increases, so that surface match verification increases as well.

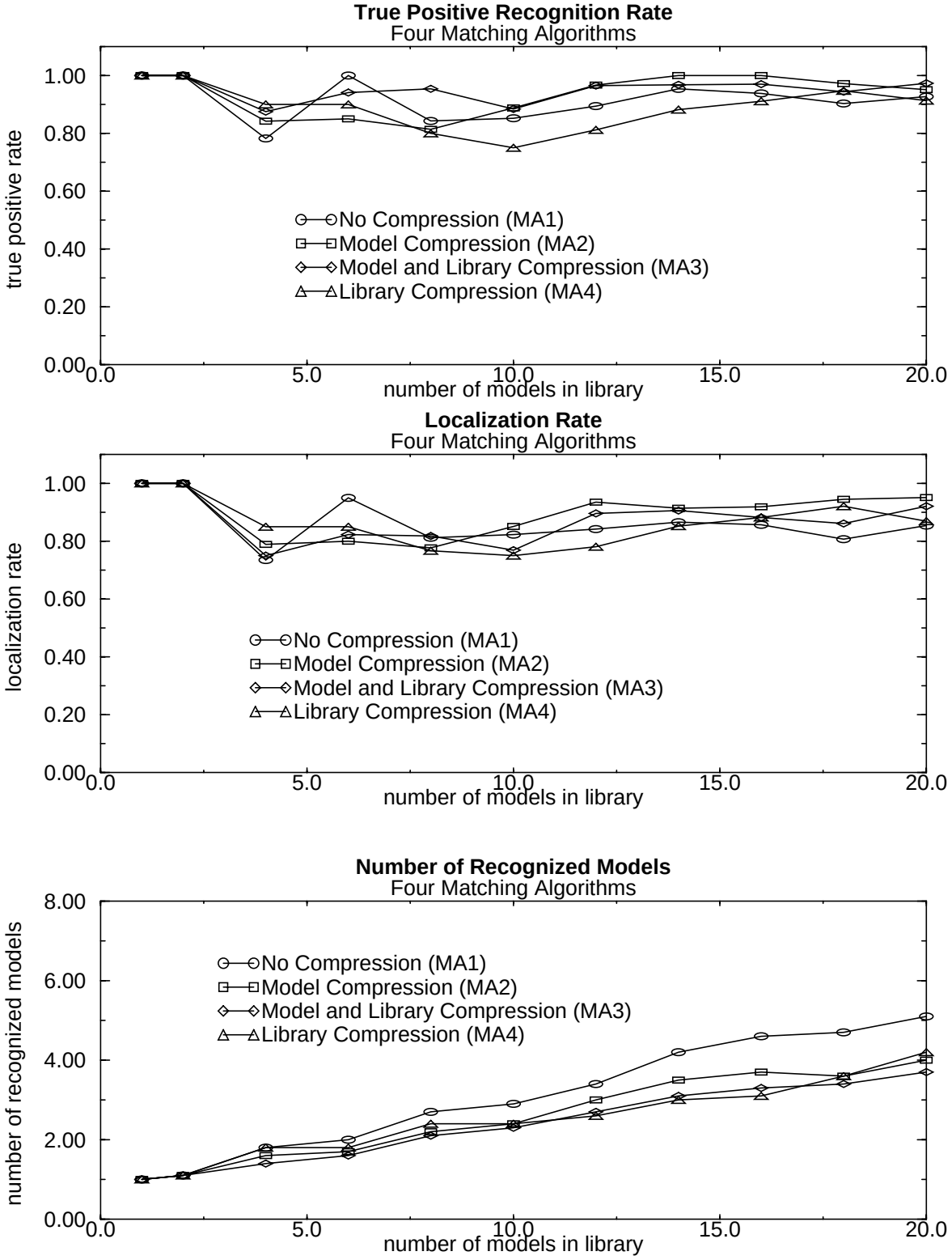


Figure 7-29: Recognition rate, localization rate and number of models recognized as library size increases for the four matching algorithms. Recognition rate (top) and localization rate (middle) remain constant as the number of models increases. Number of recognized models (bottom) increases as library size increases.

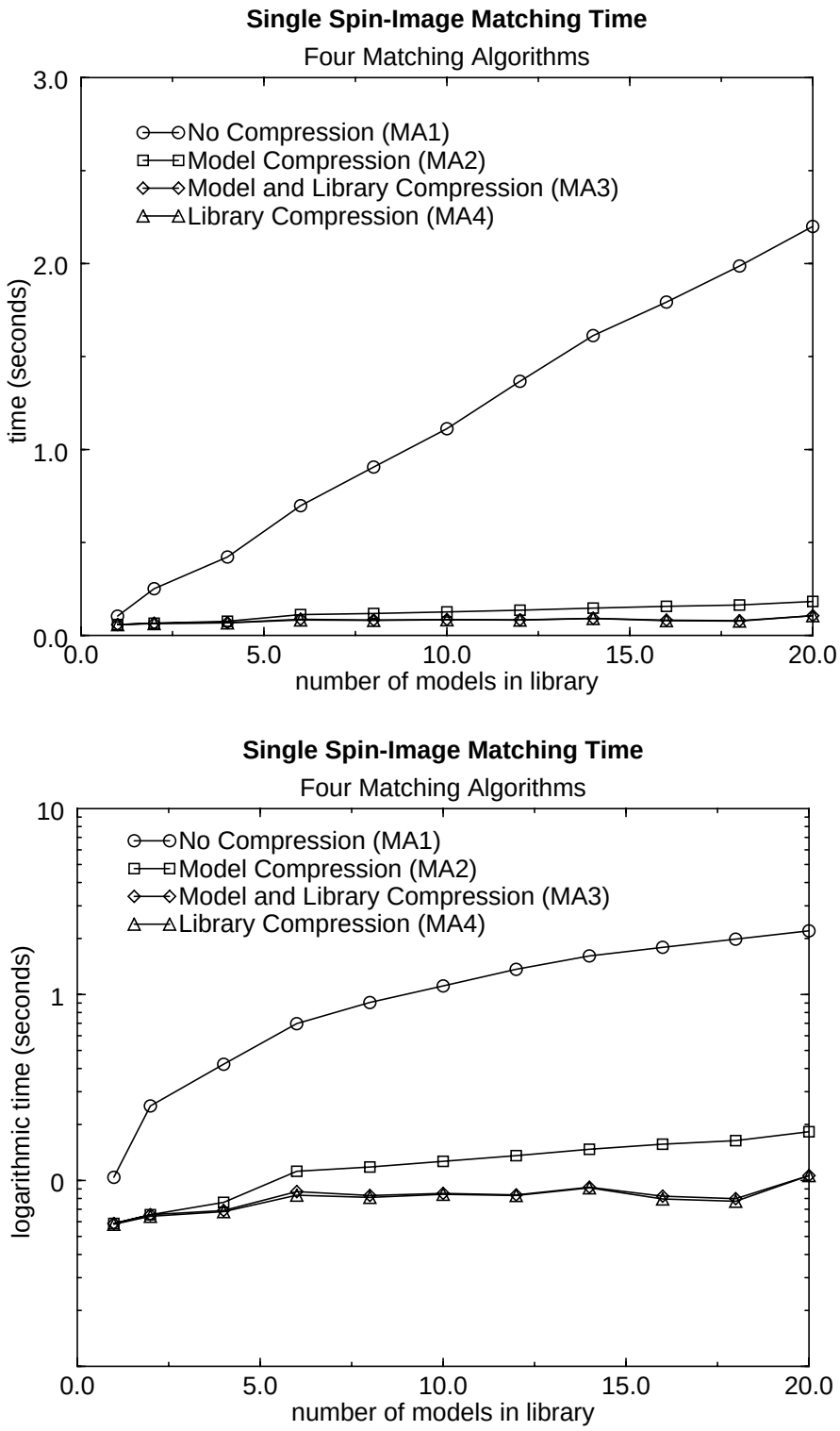


Figure 7-30: Spin-image matching times versus library size. As library size increases, spin-image matching time increases. However, the growth rate for matching without compression is much larger than the growth rate for the three algorithms that use spin-image compression. The logarithmic time plot demonstrates that for 20 models, the compressed algorithms are an order of magnitude faster than spin-image matching without compression.

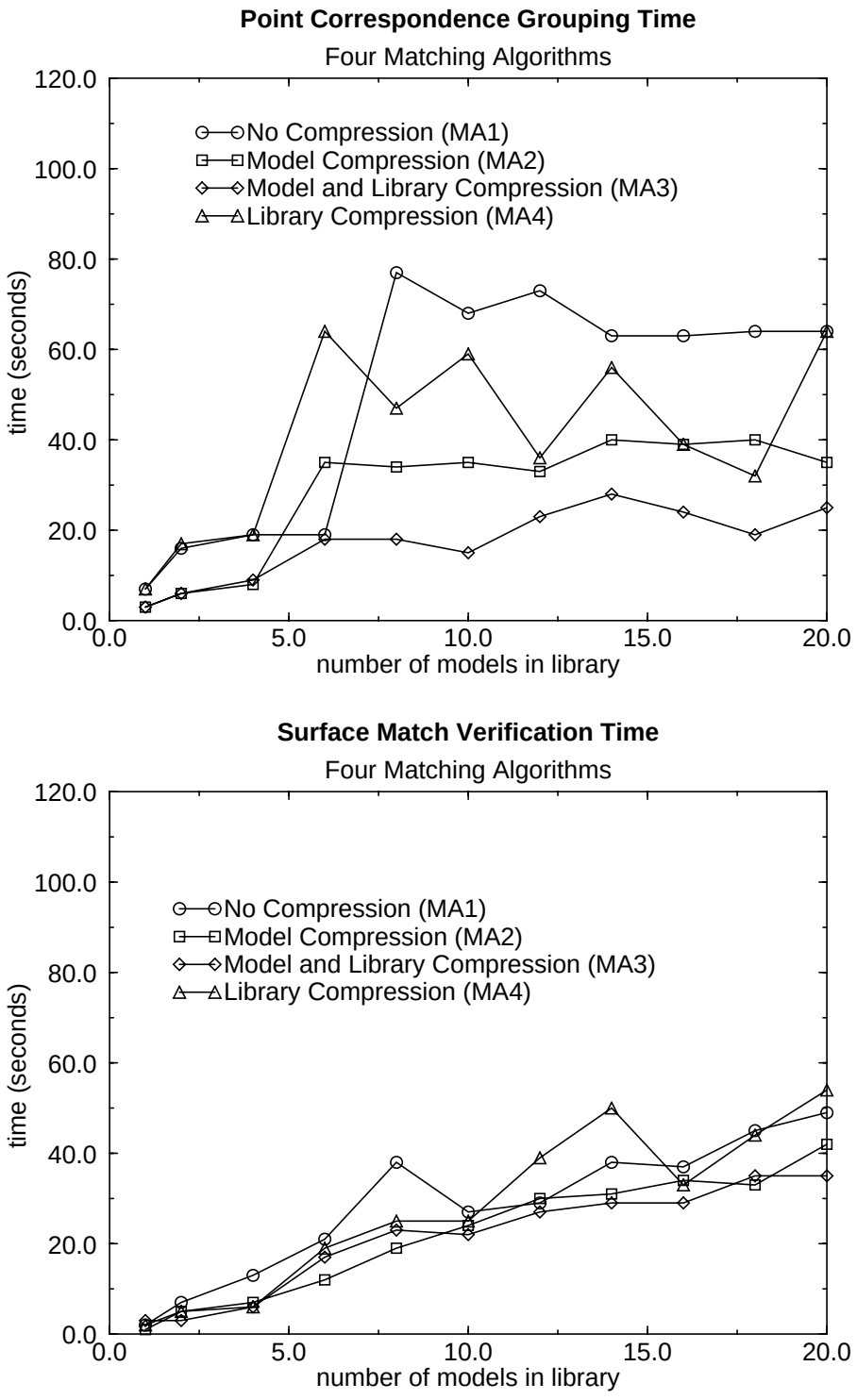


Figure 7-31: Correspondence grouping and match verification times versus library size. point correspondence grouping time (top) and surface match verification time (bottom) increase. The growth rate for matching without compression is much larger than the growth rates of the other algorithms.

Chapter 8

Application: Interior Modeling

This chapter describes *Artisan*, an interior modeling system that semi-automatically builds 3-D models of a robot's workspace. Range images are acquired with a scanning laser rangefinder and then processed, based on a systematic sensor characterization, to remove noise and artifacts. Complex 3-D objects represented as surface meshes are subsequently recognized using spin-images and inserted into a virtual workspace. This graphical virtual workspace is then used by human operators to plan and execute remote robotic operations. Artisan demonstrates the use of spin-images for recognizing objects to solve a real world problem.

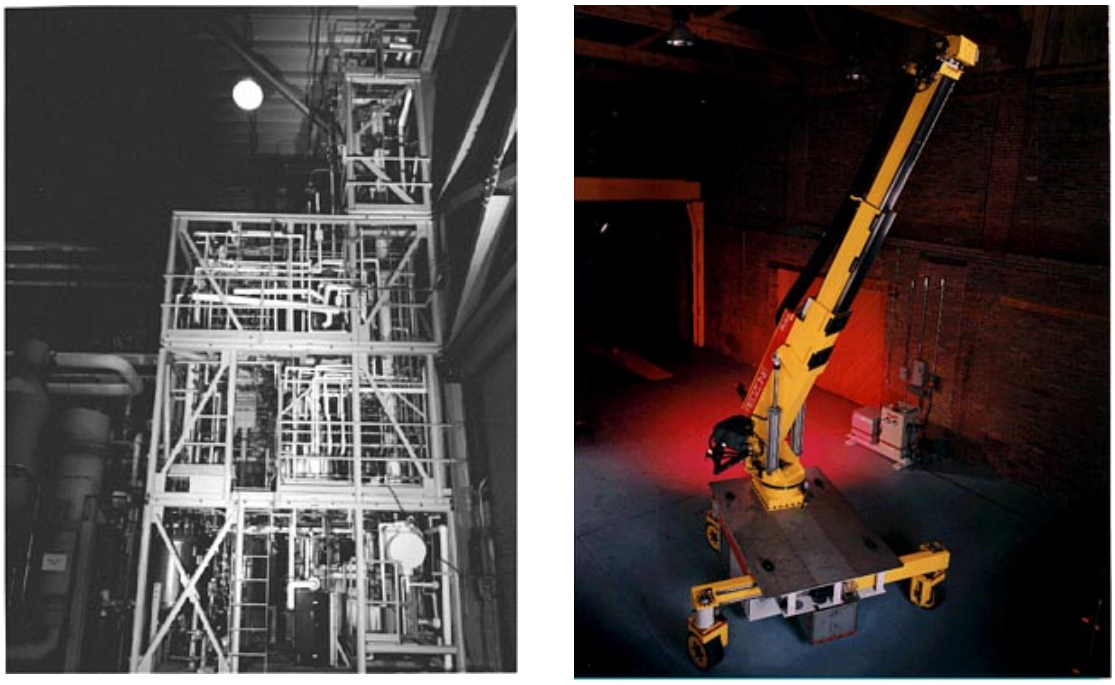


Figure 8-1: In interior modeling, a 3-D model of a complex industrial facility (left) is built to aid in its dismantlement or inspection by a remote mobile worksystem (right).

8.1 Motivation

Artisan is a system that combines 3-D sensors, object modeling and analysis software, and an operator interface to create a 3-D model of a robot's work area. This paradigm, known as task space scene analysis, provides a much richer understanding of complex, interior work environments than that gleaned from conventional 2-D camera images, allowing an operator to view the work space from inaccessible angles and also to plan and simulate robotic actions within the virtual world model [3] [52]. Through object recognition, Artisan assigns semantic meaning to objects in the scene, facilitating execution of robotic commands, drastically simplifying operator interaction and setting the stage for automatic task execution.

Figure 8-1 shows an example of a process control facility that the Artisan system is designed to model. The facility is composed of man-made objects with parametric descriptions (e.g., pipes, tanks and valves). To demonstrate the utility of interior modeling for dismantlement us

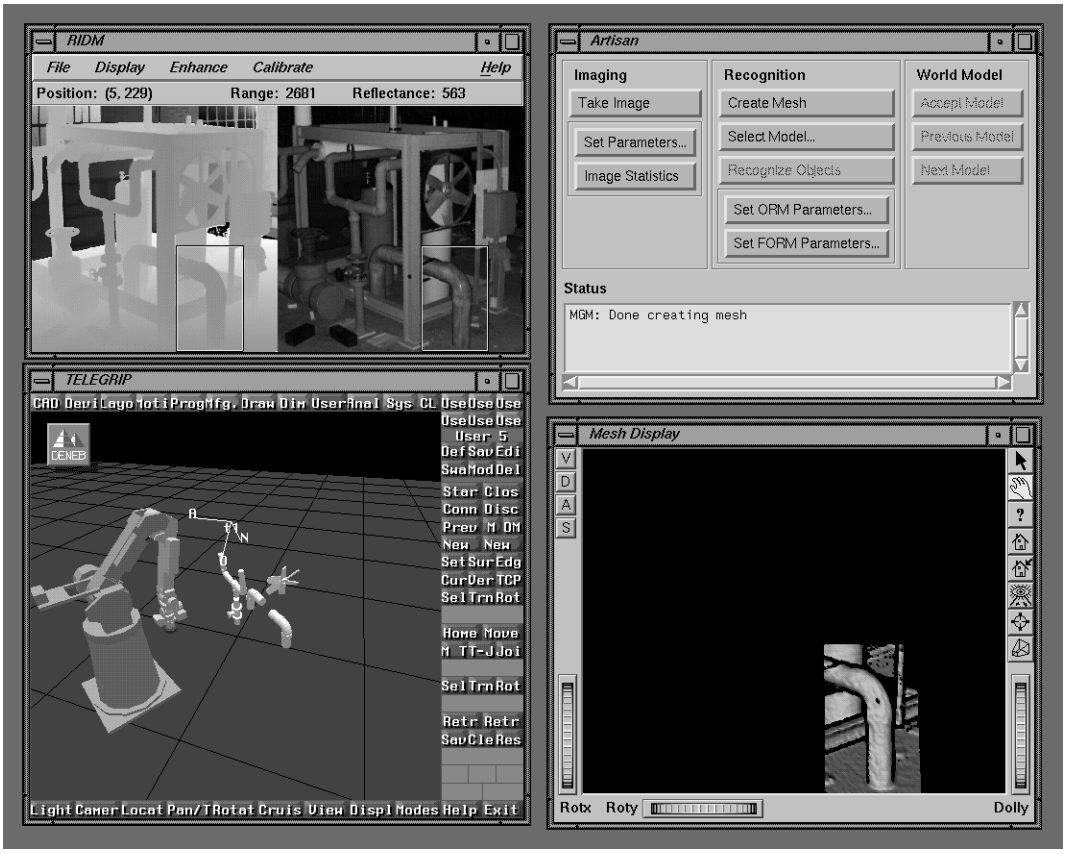
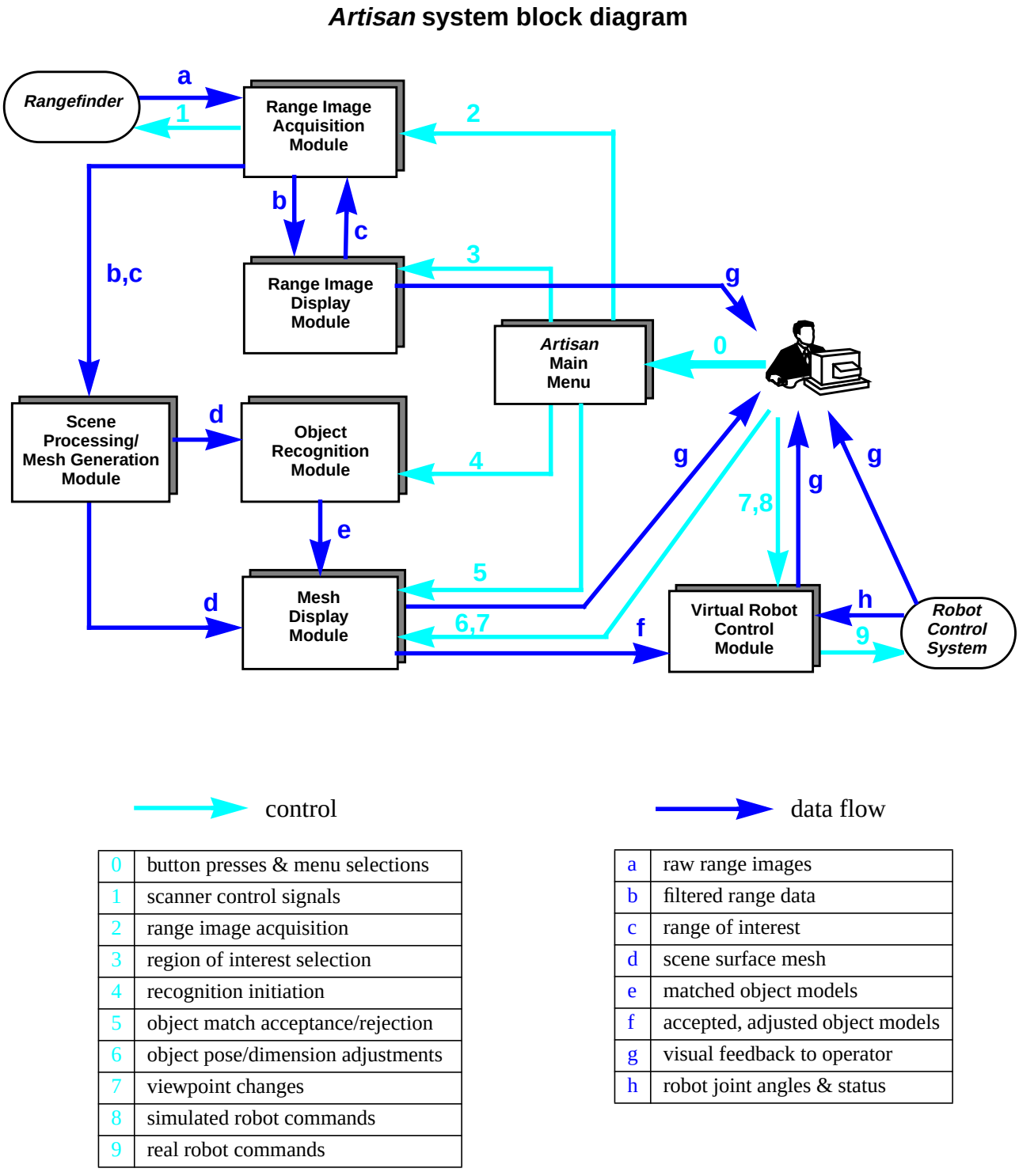


Figure 8-2: User interface for the Artisan system. Artisan consists of a control panel (upper right), a range image display module (upper left), a 3-D mesh display module (lower right) and an interface to a virtual robot world.



0	button presses & menu selections
1	scanner control signals
2	range image acquisition
3	region of interest selection
4	recognition initiation
5	object match acceptance/rejection
6	object pose/dimension adjustments
7	viewpoint changes
8	simulated robot commands
9	real robot commands

a	raw range images
b	filtered range data
c	range of interest
d	scene surface mesh
e	matched object models
f	accepted, adjusted object models
g	visual feedback to operator
h	robot joint angles & status

Figure 8-3: The Artisan scene modeling system consists of several modules for 3-D data acquisition: scene data processing, object recognition, data display and robot control. The diagram details the data flow and the necessary human interaction to make Artisan work.

ing a real robot, the Artisan system has been integrated with the Rosie Remote Worksystem [15] (shown on the right in Figure 8-1).

World modeling with Artisan proceeds as follows. From a remote workstation, the operator directs a scanning laser rangefinder to acquire images. Positioned at the work site by a mobile worksystem, the rangefinder sends range and intensity images to the operator for display. The operator then defines a desired region of interest and selects from a CAD model database objects that appear in the region of interest. From these clues, Artisan recognizes and locates the objects selected by the operator. Finally, the operator accepts or rejects models recognized by Artisan. Once accepted, the objects appear in a virtual world model at the calculated location. This process of range data collection, processing, and user interaction continues until all desired objects appear in the 3-D model of the task space. With the 3-D task space model in place, the operator can now access automatic controls such as trajectory planning, collision avoidance, and scripted motion sequences to execute tasks automatically. Figure 8-2 shows the Artisan system user interface which has four parts: a main menu for controlling perception tasks, a range and intensity image display tool, a 3-D viewing tool, and virtual robot workspace used to plan and execute robotic actions. Figure 8-3 shows all of the components of the Artisan system with data flow between them and necessary user interaction.

8.2 Scene Sensing and Processing

During world modeling, an operator commands Artisan to take an image of the scene with a Perceptron Model 5000 laser rangefinder. The Perceptron 5000 returns a 256x256x12bit range image. Its field of view is 30˚, its ambiguity interval is 10 meters, and it has a nominal standoff distance of 1.5 meters. To eliminate some of the noise in the range data, multiple range images (~10) are taken and the median range value for each pixel is returned. Range data acquisition and temporal filtering takes roughly 10 seconds. After the scene data is acquired, the range and intensity images are displayed to the operator.

Next, in the display of the range image (Figure 8-4), the operator draws a rectangular region of interest around the objects to be modeled. This reduces the data to be processed to that which is important to the task at hand. Next a scene surface mesh is created from the range image in the region of interest by making each pixel a vertex and connecting vertices across rows and

columns. If the operator chooses, the amount of scene data to be processed can be reduced by sub-sampling the range image during mesh creation. Sub-sampling is appropriate when recognizing large objects without large amounts of surface detail.

Previous work with the Perceptron laser rangefinder [58] has noted problems with laser scanning technology with regard to intrinsic scanner characteristics, surface material properties and surface incidence angle. As part of the development of the Artisan system, an effort was made to characterize and understand the 3-D laser rangefinder, and to use these results to improve the scene sensing capability of the system. For example, the graph in Figure 8-6 shows the effect of surface incidence angle on the range measurement. A polished aluminum surface at fixed distances from the sensor is scanned by the laser as the surface incidence angle increases. As the incidence angle goes beyond 20 degrees, the range measurement is no longer constant. A practical implication of this result is that shiny, cylindrical objects (such as metal pipes frequently found in industrial facilities) are difficult to model correctly.

We incorporate the results of this characterization into scene data processing by removing vertices whose incidence angles (measured as the angle between mesh normal and viewing direction) are greater than a fixed threshold. Eliminating vertices with large incidence angles has the

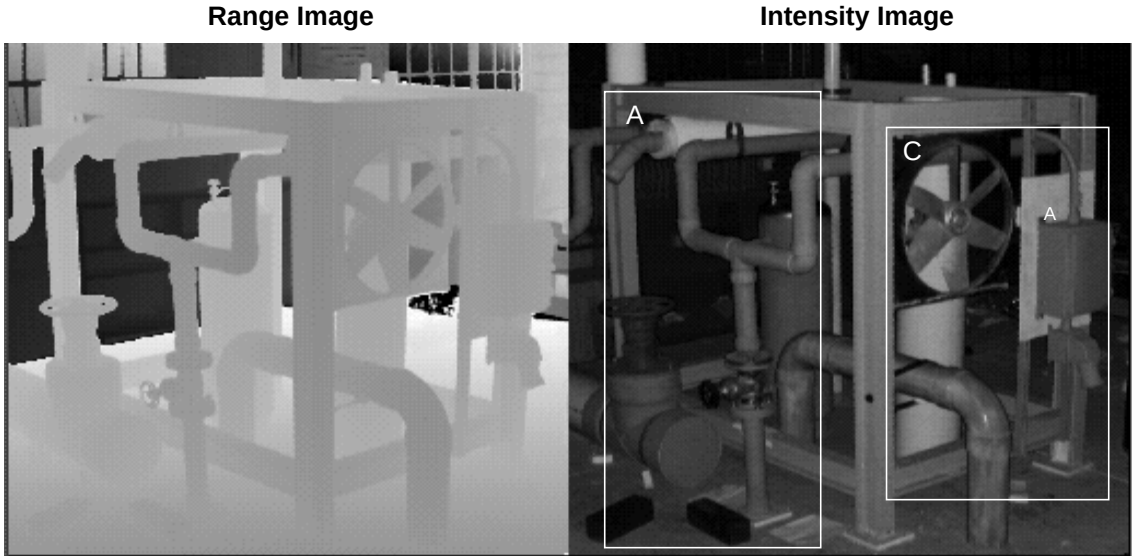


Figure 8-4: Range and intensity image with marked regions of interest. The first step in modeling the environment is to acquire a range image of the scene. Next, regions of interest outlining the location of interesting objects are drawn by the user.

added benefit of eliminating mixed pixels, pixels that lie on range discontinuities. Mixed pixels do not correspond to the points on the surfaces of objects and should be eliminated.

When a pixel is eliminated, all of the edges in the mesh that are connected to it are also eliminated. Removing edges can result in the creation of small isolated patches of pixels. Since these patches generally do not contribute to recognition, they are eliminated from the mesh to reduce the amount of scene processing for recognition.

The next step in scene data processing is the application of a low-pass filter to the mesh [88]. This filter smooths without shrinking, so it removes spurious noise from the mesh while still preserving the shape of objects. Since the filter operates on a 3-D surface mesh, and not on the range image, the smoothing provided will not contain artifacts caused by the parameterization of the range image. The final step in mesh processing adjusts the resolution of the mesh to meet the resolution of the models in the model library using the algorithm for control of mesh resolution described in Appendix A.

Figure 8-4 shows a shaded view of a scene surface mesh from a region of interest encompassing the entire range image shown in Figure 8-4 before and after processing. Scene processing for a typical region of interest of 5000 points takes approximately 20 seconds on an 100MHZ Silicon Graphics Indigo2.

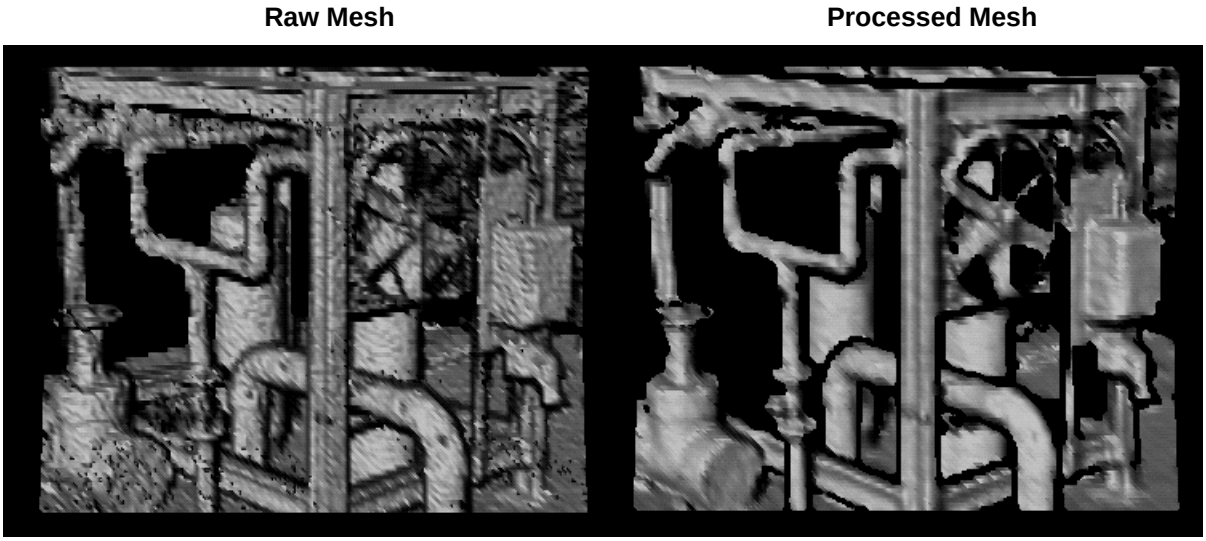


Figure 8-5: Shaded views of a scene surface mesh before (top) and after (bottom) processing. Processing removes pixels on range discontinuities, removes small patches and low-pass filters the data.

8.3 Model representation

Prior to object recognition, models of objects that are to be recognized must be created. Since we are interested in modeling complex scenes, the representation we choose must be flexible. A polygonal surface mesh is an established way to describe the shape of complex objects in computer graphics, and it is amenable to our recognition algorithm which requires a surface represented as oriented points (3-D points with associated surface normal). The vertices of a surface mesh correspond to points on the surface of the object. The normals at the vertices can be calculated by fitting a plane to the vertex and all of the neighboring vertices in the mesh. Using surface meshes places very few restrictions on the shape of objects that can be represented, making our recognition system extremely flexible.

An important requirement of any recognition system is the ability to generate the model representations used in recognition with relatively little effort. Given a CAD drawing of an object to be recognized, surface mesh generation is simple. The CAD drawing is imported into a CAD package with finite element capabilities (e.g., ProEngineer). The finite element software is then used to automatically tessellate the surface of the object into triangular faces, given some user defined constraints on minimum and maximum edge lengths. Figure 8-7 shows an example of a CAD model of a fan transformed into a surface mesh and Figure 8-8 shows some of the sur-

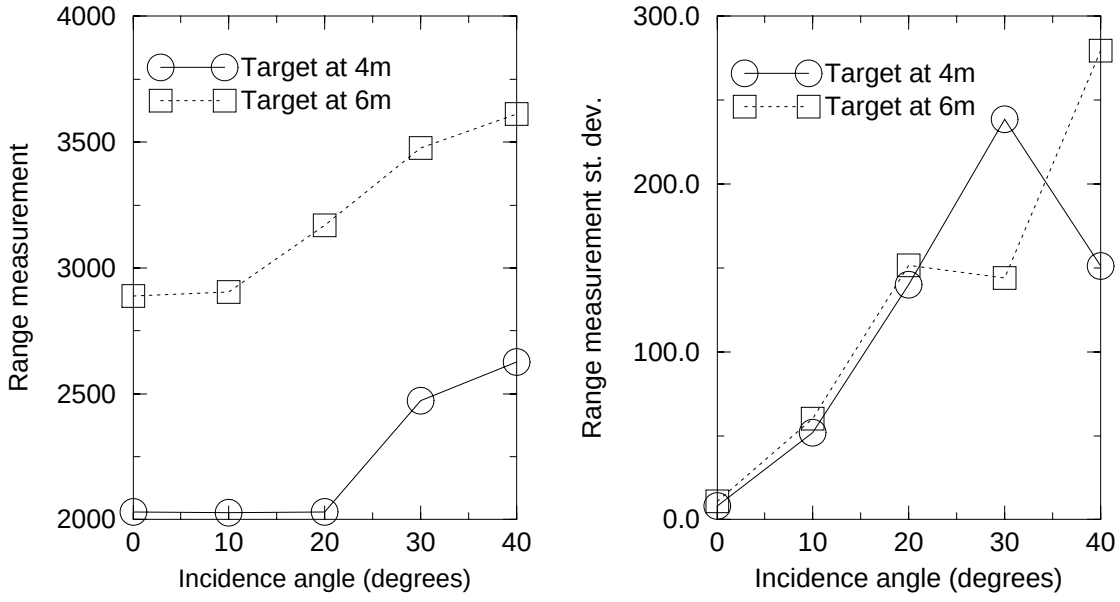


Figure 8-6: Range measurement mean (left) and standard deviation (right) for a sequence of 100 measurements of a polished aluminum target (at 4m and 6m distances) as the incidence angle is increased.

face mesh models created from CAD drawings used in Artisan's model library. Since models for recognition can be easily generated from CAD drawings, Artisan can be quickly modified to recognize objects in any man-made environment, provided the CAD drawings of those objects are available.

If CAD drawings of the objects to be recognized are not available, the object modeling algorithm described in Chapter 4 can be used to create object models.The ability to generate models of objects from range images has some useful consequences when modeling scenes with unexpected or unknown objects. If an unknown object that is important, but is not in the model library is encountered when modeling, the operator can make a model of the object simply by taking an image of the object and selecting a region of interest that contains only object data. The model representation (albeit one view) can then be created from the scene data. The model is then inserted into the model library for recognition at a later time. If a complete model of the object is desired, then multiple views of the object can be taken and registered and integrated as explained in Chapter 4. This ability to generate models at run-time is important when modeling interiors as built.

The representation we use for recognition assumes that the vertices in the surface mesh adequately describe the shape of the object. Implicit in this is the assumption that the vertices of the surface mesh are evenly distributed over the surface of the object. Occasionally, the model surface mesh generation process (FEM or Multi-view) will distribute vertices that are either too

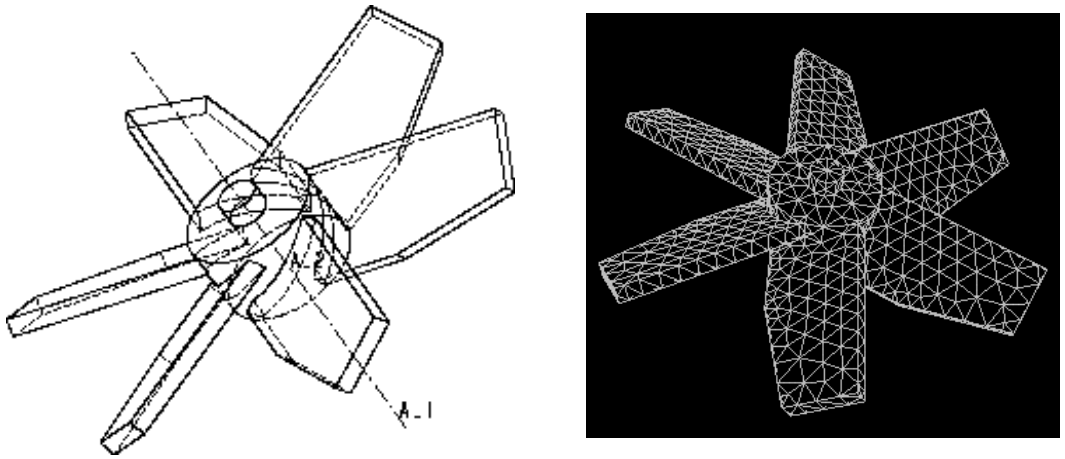


Figure 8-7: The generality of our recognition system is demonstrated by its ability to use surface meshes generated from CAD models using finite-element surface tessellation software.

coarse or too fine for recognition. To solve this problem we use a mesh simplification algorithm that normalizes the lengths of edges in a mesh while preserving the shape of the object represented [49]. The end result is a surface mesh representation of appropriate resolution that has an even distribution of points over the surface and from which our recognition representation can be made.

8.4 World Modeling

Artisan recognizes objects by matching spin-images as described in Chapter 5. During recognition, the human operator is used to speed up and safeguard recognition. After selection of the region of interest, the operator selects a model(s) from the model library to be recognized and localized in the scene. Since the model library is created at run-time, recognition using pre-generated model libraries is not possible. Therefore, only spin-image matching using uncompressed spin-images (MA1) or model compressed spin-images (MA2) is possible. After spin-image matching, the results of the recognition run are presented to the operator in a 3-D viewer window that provides rotation, translation and zoom capabilities as well as several modes of rendering. The user makes the determination of whether the recognized objects are localized well enough to be inserted into the model of the robot's world. This confirmation step affords a final human check on the validity of the object recognition.

To test our recognition system, a mock-up of an industrial control facility was built. The mock-up is composed of PVC pipes connected by joints of different angles, wooden I-beams, metal valves, a fan and a metal control box. Figure 8-9 shows the recognition of four objects in the

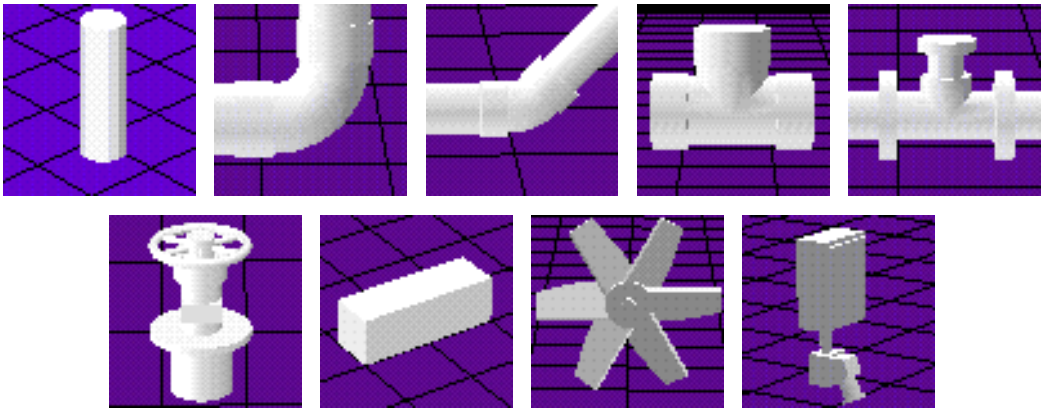


Figure 8-8: Models generated from CAD drawings used in Artisan.

mock-up from regions of interest that contain extensive clutter. Once the user accepts Artisan's results, the objects are inserted into the global 3-D world model that is captured using TeleGRIP (by Deneb Robotics, Inc.).

Once the objects are placed in the workspace, robotic tasks can be planned and executed virtually before being executed by the real robot. This provides safer, more reliable execution of remote tasks than is possible in teleoperation using conventional 2-D cameras. Furthermore, because a semantic meaning is attached to objects in the virtual world, high level commands to the robot can be used to perform tasks (e.g., "cut that pipe", "turn that valve").

The Artisan system has been demonstrated in multiple dismantlement and decommissioning contexts. In a decontamination task using a T3 manipulator, for example, Artisan correctly identified and located objects in the manipulator's work space, and a simple trajectory to follow surfaces of pipes and vessels in the workcell was computed using TeleGRIP's motion planning features. The operation was executed first in simulation; then the proper commands were downloaded to the real manipulator's controller, which then performed the wash-down operation. In this manner, locations of the real objects could be compared to the locations of the recognized objects. Observed deviations were on the order of 1-4 cm which demonstrates that localization of objects using range data and spin-image matching is accurate enough for typical dismantlement operations.

Artisan has also been integrated into the Rosie mobile worksystem to enable semi-automatic size reduction of the lower shield plug of the Argonne National Lab CP-5 experimental reactor. Using the observed motions of the real Rosie relative to the shield plug (actually a mock-up), overall task performance in this case had lower fidelity due, in part, to the scale of the object (approximately ten feet in diameter) relative to the laser rangefinder's field of view (30˚). The lower fidelity is also attributable to the somewhat lower stiffness of Rosie's heavy manipulator and controller with respect to the T3 manipulator.

8.5 Discussion

Our approach to object recognition offers substantial improvements in performance over previous systems. First, the approach does not make any assumptions on the sensor used for ac

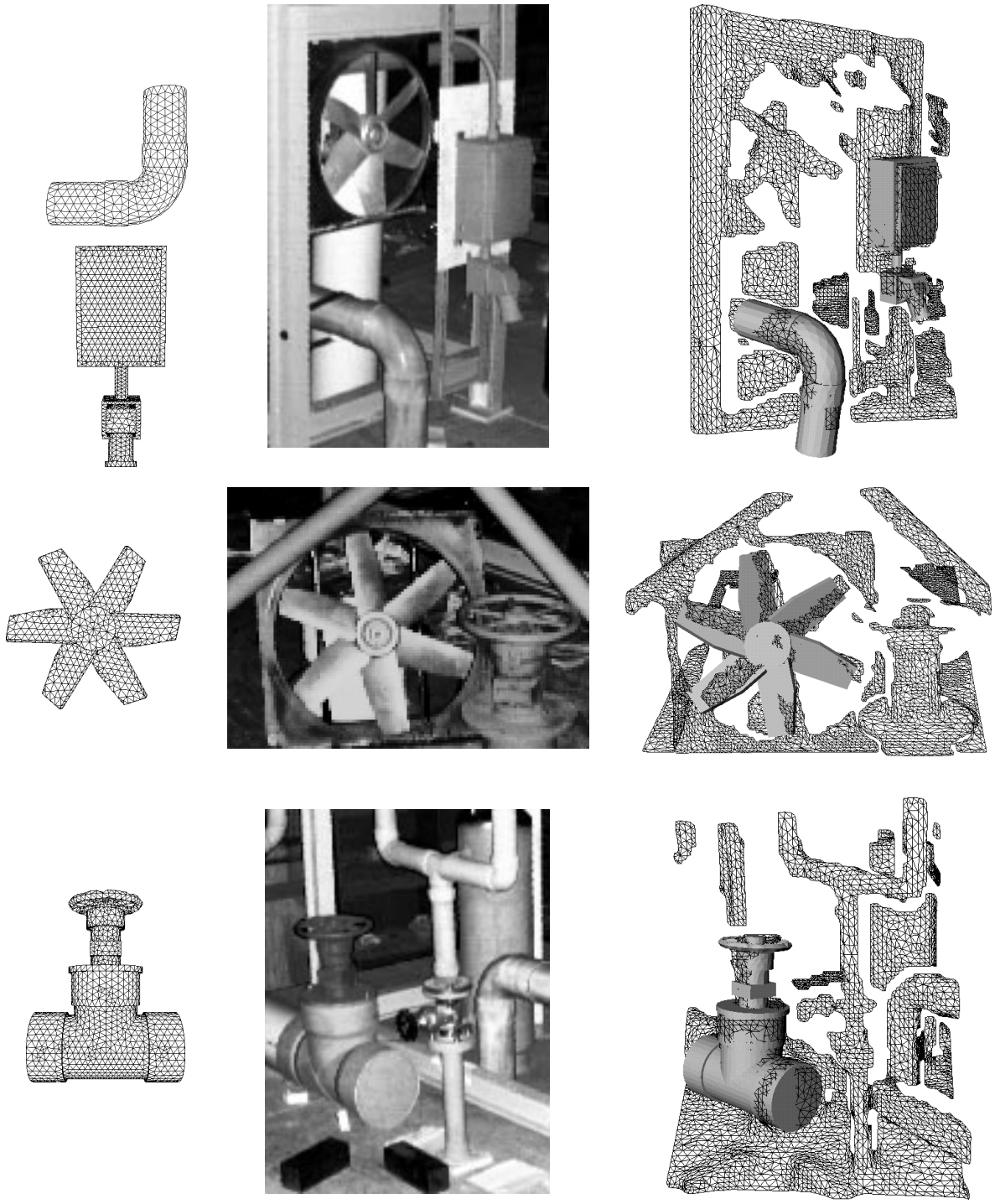


Figure 8-9: The recognition of industrial objects in complex scenes. On the left are shown wireframe models which were created from CAD drawings using finite element software for surface tessellation. In the middle are shown the intensity images acquired when a scanning laser rangefinder imaged the scene. On the right are shown the recognized models (shaded) superimposed on the scene data (wireframe). These results demonstrate the recognition of complicated symmetric objects in 3-D scene data containing clutter and occlusions. The top and bottom results come from regions of interest A and C shown in Figure 8-4.

quiring the range data. In fact, we have demonstrated the system both with an imaging laser rangefinder and with a light stripe rangefinder. In contrast, systems based on interactive stereo [90] or line segmentation make strong assumptions on the geometry of the sensor and do not generalize nearly as well as does the point matching algorithm at the heart of Artisan.

Another drawback of competing techniques is that they tend to degrade the input range data by forcing it into one of the primitive classes. Our approach is general in that, by working directly with surface meshes, it does not make assumptions of the shape of the objects that can be recognized. Our earlier system [52] restricted recognition to objects mostly composed of planar and quadric surfaces. This constraint relegated Artisan to use in environments comprised only of objects with simple geometries, such as cylinders and boxes. Further, our earlier system's ability to recognize was much more dependent on the amount of range data available for a given object in the scene. This dependency made the system more susceptible to the occlusions that are prevalent in industrial process complexes; the earlier Artisan could not determine that two segments of a horizontal pipe lying behind a vertical pipe were actually the same object. In addition, the 3-D segmentation step is computationally expensive. The newer Artisan can do everything its predecessor did in about half the time.

Artisan's ability to model objects, even when surfaces are partially occluded or the sensing viewpoint changes, makes it a valuable tool for modeling complex environments and planning remote robotic tasks. Artisan is effective because user input is used to make high level decisions for the system while low level recognition and registration are done automatically. Furthermore, Artisan was designed with generality in mind: the system can work with any sensor that acquires 3-D surface data; object models can be generated easily from CAD drawings; and models of new unknown objects can be built and incorporated into the system at run-time.

Another limitation is that the object models are represented at a single scale. For example, we can recognize a pipe and its position in the image but we cannot compute its radius if it deviates too much from the model's radius. One approach is to parameterize the spin images as functions of the scale of the objects. We are investigating this possibility.

Chapter 9
Analysis of Spin-Images

During spin-image generation, the user sets parameters that control the construction of spin-images and their effectiveness in matching. After becoming familiar with spin-images, some rules of thumb for setting parameters become apparent. For example. Increasing the number of pixels in a spin-image increases its descriptiveness, or setting the bin size of spin-images to approximately the resolution of the mesh is necessary for making spin-images from models with different surface sampling similar. However, rules-of-thumb are not always sufficient for intelligently choosing parameters during spin-image generation. Therefore, in this chapter we present a detailed experiment analysis of the effects of spin-image parameters on the descriptiveness of spin-images and spin-image matching. The analysis describes the effects of varying different parameters on matching of spin-images and explains the reasons for the effects in light of the spin-image generation process. This detailed analysis also shows the generality of the spin-image representation; through intelligent choice of parameters, spin-images can be tuned for specific surface matching applications.

The parameters that control spin-image generation are bin size, image width, support distance and support angle. These *generation parameters* control how descriptive spin-images are, independent of matching. Section 9.1 first describes the statistics, termed *generation statistics,* used to analyzes the descriptiveness of spin-images. Next, it presents an experimental analysis showing how generation statistics vary as parameters are varied.

During surface matching, differences between model and scene properties affect matching. For example, mesh resolution, surface sampling and noise are all properties that affect matching. These properties are important when matching two surfaces and cannot be controlled during

spin-image generation, so they are termed *matching variables*. Section 9.2 first describes the *matching statistics* that convey the performance of spin-image matching between two models. It then shows how matching statistics vary as matching variables are changed.

During analysis of generation parameters and matching variables, results are presented for the three different models shown in Figure 9-1. The duck model represents a closed object of genus zero with a fairly simple shape typical of the objects in our object recognition system. The femur model represents an open object of slightly more complicated shape than the duck model. The range image model is even more complicated and is typical of range views of scenes used in object recognition. These three objects were chosen in order to sample the space of possible shapes from which spin-images will be generated. To adequately analyzes the effect of parameters on generation and matching of spin-images, the space of all possible shape models would have to be explored. Unfortunately, an exhaustive analysis using all possible shapes is not computationally feasible, and generating an even distribution of possible shapes is still an open research issue. Fortunately, the agreement in analysis between the different shapes chosen shows that the analysis is, for the most part, independent of shape.

9.1 Spin-Image Generation Parameters

Spin-images generation is affected by four parameters. Bin size determines the geometric size of the bins used to accumulate points. Image width determines the width of the spin-image in pixels or bins. Support distance is a geometric measure of the sweep volume for a spin-image. These first three parameters are dependent because support distance is defined as the bin size

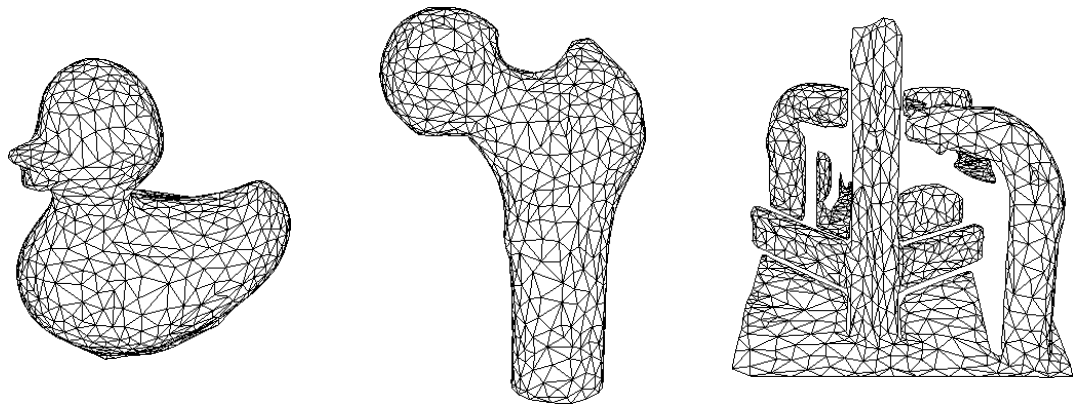


Figure 9-1: Models used in analysis.

times the image width. The final parameter is the support angle. All of these parameters affect the descriptiveness of spin-images. The following sections detail the metrics used to measure spin-image descriptiveness and then give an analysis of each parameter.

9.1.1 Generation Statistics

Generation statistics describe the spin-images from a model generated using a specific set of parameters. By plotting generation statistics varying a single parameter, while keeping the other parameters fixed, the effect of the parameter on the spin-image generation can be determined.

Since surfaces are matched by matching points, the correctness of hypothesized point matches between model and scene is important. A statistic that measures the correctness of point matches is determined as follows. First, assume that the model and scene being compared are already registered; since this is analysis, and not recognition, this assumption is reasonable. Next, establish feasible model-scene and scene-model point matches by matching spin-images as follows. For each point in the model, find its best corresponding points in the scene by comparing spin-images (Section 2.4) and determining outliers in the similarity measure histogram (Section 3.1). Similarly, for each point in the scene find the best corresponding points in the model. Store these hypothesized matches in a list. For each match, compute the distance between the position of the scene point and the position of the matching model point. Histogram these distances and compute the median. Since the model and scene are registered, the median of the histogram will be small when the point matches are correct, and large when the point matches are incorrect. The median is a robust way to condense the distance between point matches into a single number because incorrect matches with large match distances will have no effect on the statistic. We call the median of match distances the *match distance statistic*.

The match distance statistic can be computed for a single model by comparing the model to itself. Each spin-image in the model is compared to the other spin-images in the model and, as above, point matches are established by finding outliers in the similarity measure histogram. The median of the histogram of distances between corresponding points in the matches is used as the match distance statistic. If nearby points on the model are matched, then the match distance statistic will be small; if the matches are created between points that are far apart, then

the match distance statistic will be large.

Some auxiliary statistics that help explain the reason for changes in the match distance statistic are the *match overlap statistic* and the *match correlation statistic*. The match overlap statistic, is the median of the histogram of the overlap between spin-images computed for all of the point matches. Since overlap controls the confidence in correlation coefficient, it will influence the discrimination between spin-images. The match correlation statistic is the median of the histogram of the correlation coefficient between spin-images computed for all of the point matches. Correct matches are established when correlation is high. Taken altogether, a model will have discriminating spin-images when the match distance statistic is low, the match overlap statistic is high and the match correlation statistic is high.

Another statistic, unrelated to the match statistics, is the number of significant eigen-spin-images for the model, termed the *model dimension*. This statistic is computed for a model using the technique described in Section 6.2 for a reconstruction level of 95%. When the model dimension is low, the spin-images are very correlated, so they do not contain a lot of information; differences between spin-images will be hard to detect, and incorrect matching will follow. A low model dimension occurs when the number of occupied pixels in a spin-image is small and is therefore related to the match overlap statistic. When the model dimension is high, the set of spin-images from a model are less correlated, so discriminating between spin-images will be easier. In general, a high model dimension will occur when the overlap between spin-images is large and the correlation of spin-images from different parts of the model is low. When comparing the model dimension between models, models of higher complexity will generally have higher model dimension.

The following sections describe the effects of the four generation parameters in terms of the four statistics mentioned above. Before analysis, the models are resampled to 1000 points using the algorithm described in Appendix A. Then the meshes are scaled so that the resolution of each surface mesh is 1.0, enabling the variability of parameters that depend on mesh resolution to be compared between the analysis models.

9.1.2 Image Width

Assuming square spin-images, the image width is the number of pixels in a row or column of

the spin-image. As the image width increases, the total number of pixels in the spin-image (i.e., its dimension) goes up quadraticly. Intuitively, increasing image size will increase the uniqueness of spin-images, making them more discriminating during matching. We created an experiment to validate our intuition and to motivate discussion of the role of image width in matching. Using the resampled and scaled models, we calculated the four generation statistics using spin-images generated such that the image width varied from 2 pixels to 20 pixels while the bin size was fixed at twice the model resolution and the support angle was fixed at 180 degrees (to give maximum amount of information in the spin-images). Since the image width varied while the bin size was fixed, support distance increased linearly with image width. The effects of image width on the four generation statistics are shown in Figure 9-2 and Figure 9-3.

For maximum discrimination, image width should be set as large as possible.

The match distance statistic clearly decreases as image width increases until an image width of 10 pixels where it begins to level off. At low image width, the spin-image support distance is small and surface area projected into the spin-image is small. Therefore, the spin-images will contain fewer shape features and mismatches will occur, causing the match distance statistic to be large. At large values of image width, the match distance statistic levels off at a value close to the resolution of the surface meshes (1.0). Since the mesh resolution places a lower bound on the distance between closest points, this indicates that points that are close to each other are being matched. In addition to validating that large values of image width result in better matching, the best match statistic also shows that there is a point of diminishing returns where increasing the image width (and consequently increasing computation) does not result in significant improvements in spin-image matching. Of course, these conclusions are relying on bin size being set to an appropriate value; setting bin size is discussed in Section 9.1.3. Another consideration not discussed in this analysis is that, to reduce the effect of clutter and occlusion on surface matching, the image width should be kept small. In applications where it is necessary, striking a balance between these forces is necessary for effective matching.

To fathom the reason why the match distance statistic improves with increasing image width, the match overlap and match correlation statistics can be consulted. Match correlation starts, and remains, at a high value as image width increases. On the other hand, match overlap starts out small and increases as image width increases. As previously mentioned, the increase in

match overlap is due to the increase in support distance as image width is increased. Since the overlap between best match spin-images is increasing, the information used to discriminate between images is increasing, so better matching occurs. The slight decrease in match correlation as image size increases is due to more pixels being used in the computation of correlation coefficient, resulting in a lower correlation coefficient for matches, but a higher certainty that the correlation coefficient is correct.

As image width increases, the model dimension also increases. This is because the dimension of the spin-images is increasing and the match overlap is increasing. As image width increases, the spin-images are transforming from a very local representation where matching is less discriminating, to a global representation where images encode the overall shape of the model, causing variation between spin-images to increase. In the case of the duck and femur models, the model dimension levels off around an image width of 15. This is because the support distance has increased to around the size of the models. Increasing the image-width beyond this value will result in little additional surface being included in the spin-images, so the variation between spin-images will not change and the model dimension will level off. This conclusion is validated by the match overlap statistic, which also shows the match overlap leveling off for the duck and femur models around an image width of 15.

The difference in complexity between the three models is born out by the match distance statistic and the model dimension. Visually, the models can be ranked in complexity; the range model is the most complex followed by the femur and then the duck model. This complexity order is repeated in the match distance statistics once they level off. The range model has the smallest match distance, followed by the femur and then the duck. Increased model complexity results in increased discrimination between spin-images and a lower match distance. The model dimension is higher for more complex models for similar reasons. (Describing model complexity in terms of shape similarity is future work and is discussed in Chapter 10.)

9.1.3 Bin Size

Bin size is the geometric size of the bins in the spin-images generated. The larger the bin size, the more averaging occurs in the spin-images. The rule-of-thumb is to set the bin size to roughly the resolution of the model surface mesh. We designed an experiment to test if this rule-of-

thumb is valid. Using the resampled and scaled models, we calculated the four generation statistics using spin-images where the bin size varied from one tenth the model resolution (0.1) to ten times the model resolution (10.0). The image width was fixed at 10 and the support angle was set at 180 degrees. The plots of generation statistics from the experiment are shown in Figure 9-4 and Figure 9-5. Since the image width was fixed while the bin size varied, the support distance also varied as the bin size times the image width. (An experiment where the support distance is fixed while bin size and image width vary is described in Section 9.1.4.)

Bin size should be set to approximately the mesh resolution.

The match distance statistic is large for bin sizes less than the mesh resolution, so poor matches are being produced. After the bin size passes the mesh resolution, the match distance levels off close to the mesh resolution, so correct matches are occurring. Consulting the plot of match overlap, we see that below the mesh resolution, the overlap is small, but increasing. Around the mesh resolution, the overlap levels off and then begins to decrease as bin size increases. The small overlap, at a fixed image size, below the mesh resolution is caused by the way in which points are accumulated in the spin-image. When the bin size is much smaller than the mesh resolution, different vertices in the mesh are falling into different bins of the image. Points are not being accumulated and the averaging to reduce the effects of the discrete sampling of the surface by the vertices of the mesh is not occurring. Therefore, spin-images that are close to each other will be dissimilar. This is validated by the plot of match correlation, which shows smaller correlation coefficients for bin sizes less than the mesh resolution. Since the image width is fixed, decreasing the bin size will also decrease the support radius; this will reduce the overlap as well.

Near the resolution of the mesh, multiple points are being accumulated in each bin of the spin-image, so an averaging that reduces the effect of the discrete location of vertices on the surface is occurring. Furthermore, the match overlap and match correlation are large, so spin-images are discriminating. This results in a low match distance statistic.

As bin size increases, the match overlap decreases, because the number of bins accumulating points is decreasing. Eventually, the bin size reaches the size of the model, and all the points in the model will be accumulated in the two bins closest to the origin of the spin-images.

The model dimension has a complicated plot because of different mechanisms at play. At small bin sizes, very little averaging of point positions is occurring, and the discrete nature of the surface is apparent in the spin-images are generated. This causes the set of spin-images generated for a model to span the space of spin-images, resulting in a large model dimension. As bin size increases, averaging by accumulation takes over and the spin-images become more correlated, thus reducing the model dimension. Since the support distance is increasing with bin size, more surface area will be accumulated in the spin-images as bin size increases. This eventually causes the significant dimension statistic to increase because more discriminating surface features are being stored in each spin-image. Eventually the bin size becomes large enough that the support distance is on order of the size of the model and the match overlap starts to decrease. This will cause an equal decrease in the model dimension.

Taken all together, the generation statistics indicate that an appropriate value for bin size is on order of the resolution of the mesh. This will result in large overlap and correlation, a small match distance and a large number of significant eigen spin-images. Of course, this rule-of-thumb is dependent on the image size being set large enough to convey enough of the surface shape.

9.1.4 Support Distance

Support distance is a geometric measure of the sweep volume of a spin-image. Larger support distances result in more discriminating spin-images because more surface area of the model is conveyed by each spin-image. For square spin-images, the support distance is equivalent to the sweep radius and the sweep height. During spin-image generation, the support distance is set when image width and bin size are set; support distance is the product of the image width and bin size. To illustrate the dependency of support distance, bin size and image width, we developed the following experiment. Spin-images of constant support distance were generated for the three analysis models by varying bin size and image width. The constant support distance was set to 10.0 and spin-images were generated for (bin size, image width) pairs of (1.0,20,0), (2.0,10.0), (2.86,7.0), (4.0,5.0), (5.0,4.0), (6.66,3.0) and (10.0,2.0). Plots of the generation statistics for these spin-images are shown in Figure 9-6 and Figure 9-7.

For fixed support distance as the bin size increases and the image width decreases, the match

distance statistic increases. This is caused by the decrease in the number of pixels in the spin-images which results in a decrease in match overlap and, consequently, fewer discriminating spin-images. Although the same number of points are being accumulated in the spin-image, more averaging is occurring as the bin size increases. Match correlation increases as bin size increases because, as the number of pixels in the spin-images drops and more averaging occurs, spin-images are becoming more similar.

The model dimension is large for small bin-size and large image width and approaches zero for large bin size and small image width. As above, this is due to the decrease in the number of pixels in the spin-images.

Given a fixed support distance, set image width large and bin size small.

The analysis of image width, bin size and support distance can be combined to produce a method for setting these parameters. As shown in the clutter analysis in Chapter 5, the support distance should be set based on the size of the models and the amount of clutter in the scene. If there is no clutter, then set the support distance to the size of the model; if there is clutter, then set the support radius to a fraction of the model size (in practice, one half to one third the model size works well). The bin size must be set to at least the mesh resolution. If the mesh has an uneven distribution of vertices over its surface, then the bin size should be set greater than the mesh resolution, but still as small as possible. Given the support distance and bin size, the image width is set to the support distance divided by the bin size.

9.1.5 Support Angle

Support angle is the angle between the direction of the oriented point basis of a spin-image and the surface normal of points contributing to the spin-image. Support angle is used to limit the effect of self occlusion and clutter during spin-image matching. In general, support angle should be set as large as possible given the expected amount of clutter in the scene. To test this hypothesis, we generated spin-images for the three analysis models such that support angle varied from 10 degrees to 180 degrees, while bin size was fixed at two times the mesh resolution and image width was set to 10. The plots of the generation statistics for these sets of spin-images are shown in Figure 9-8 and Figure 9-9.

The best match distance statistic decreases until it reaches 90 degrees, at which point it levels

off at around the mesh resolution. The match distance improves as support angle increases, because the match overlap increases with support angle. When support angle is small, the surface area that is mapped into each spin-image is small because points in the sweep volume with normals that are very different from the oriented point direction are not accumulated. As support angle increases, the number of points accumulated increases and the number of bins with points increases, too. When the match overlap levels off, the match distance levels off, too. The leveling off of match overlap is a direct effect of the limited sweep height of the spin-images. The half sweep height of the images is 10.0 (bin size of 2.0 time half width of 5) which is less than the nominal sizes of models (15,16 and 30). Therefore, the backsides of the models are rarely included in the spin-images, independent of the support angle. For larger sweep heights, the overlap would keep increasing until it reaches 180 degrees. For all support angles, match correlation is large; the slight decrease with increasing support angle is due to the increase of match overlap, making the correlation coefficients lower, but more reliable.

Support angle should be set as large as possible.

For the duck model and the femur model, the model dimension decreases as support angle increases. This is because the duck and femur do not have very complex shapes. When the support angle is small, there is a large variation among the spin-images because each spin image has a few occupied pixels scattered throughout the spin-image. As the support angle increases, the scattered pixels become connected and the overlap between the spin-images increases. However, the simple shape of the models causes the spin-images generated with large support angles to have little variation. Therefore, the significant dimension will decrease. The range model exhibits the expected behavior of spin-images. As match overlap increases, the variation among spin-images increases and consequently, the model dimension will increase.

From this analysis, we can draw the conclusion that the support angle should be set as large as possible in order to maximize the match overlap and increase that variation between images. However, the clutter model in Chapter 5 also places an opposing constraint; support angle should be set small to limit the effect of clutter and occlusion. Therefore, a trade-off exists. For recognition, the support angle should be kept small to allow for recognition of complete object in partial scenes; for registration of complete surfaces it should be kept large.

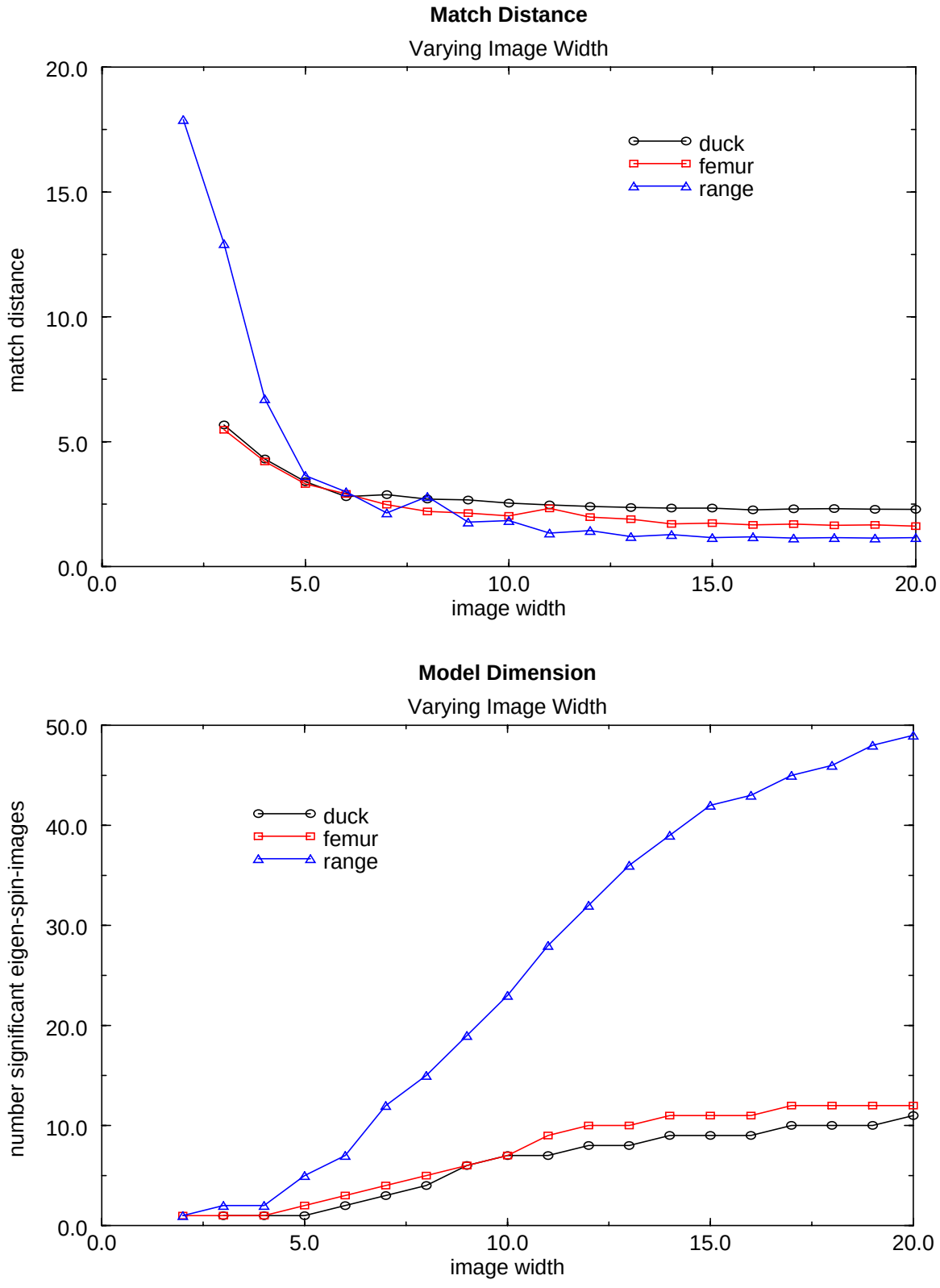


Figure 9-2: Effect of spin-image width on match distance statistic and significant dimension statistic, keeping bin size and support angle fixed at 2 times the mesh resolution and 180 degrees, respectively.

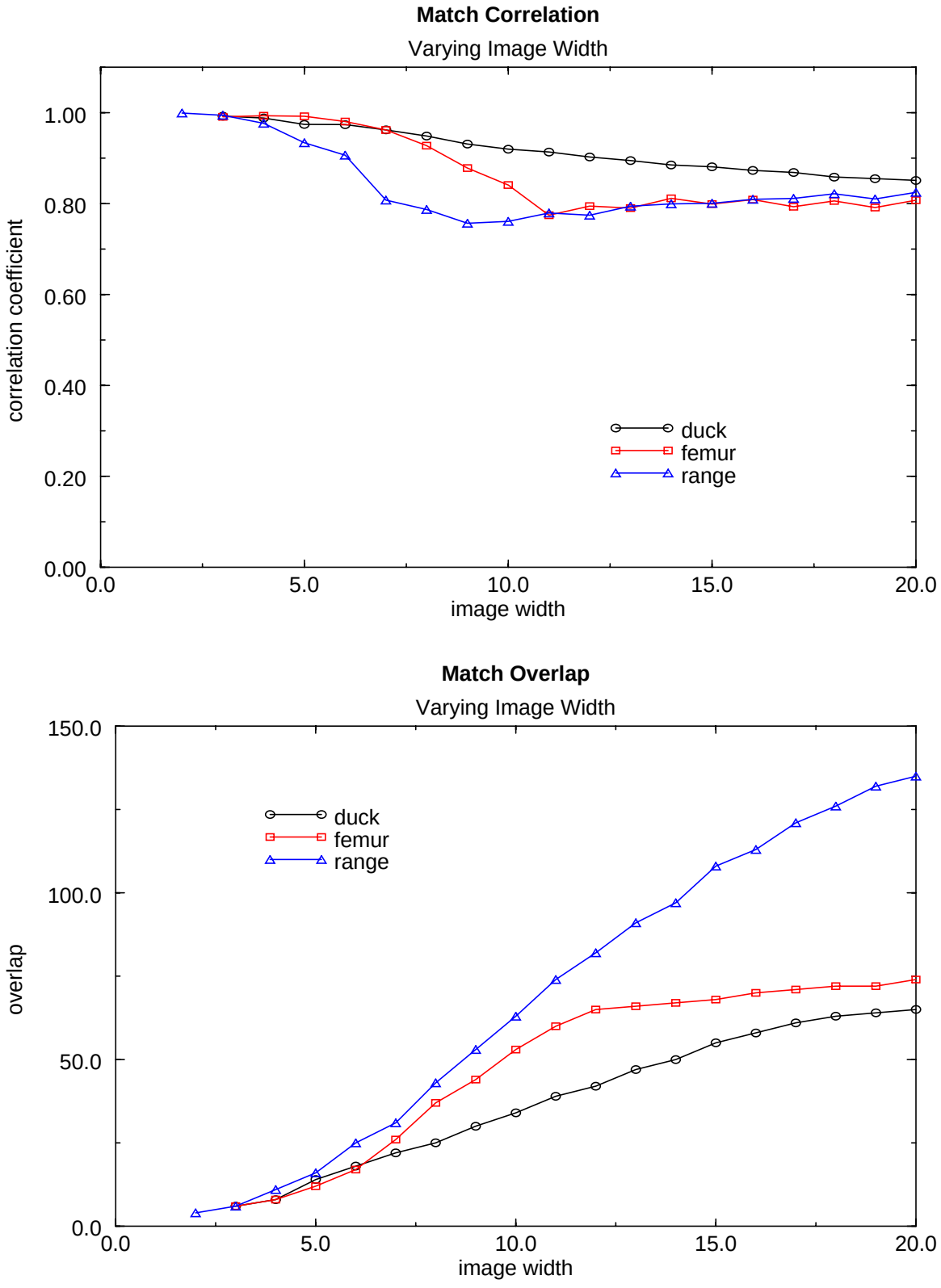


Figure 9-3: Effect of spin-image width in pixels on the match overlap and match correlation statistics, keeping bin size and support angle fixed at 2 times the mesh resolution and 180 degrees, respectively.

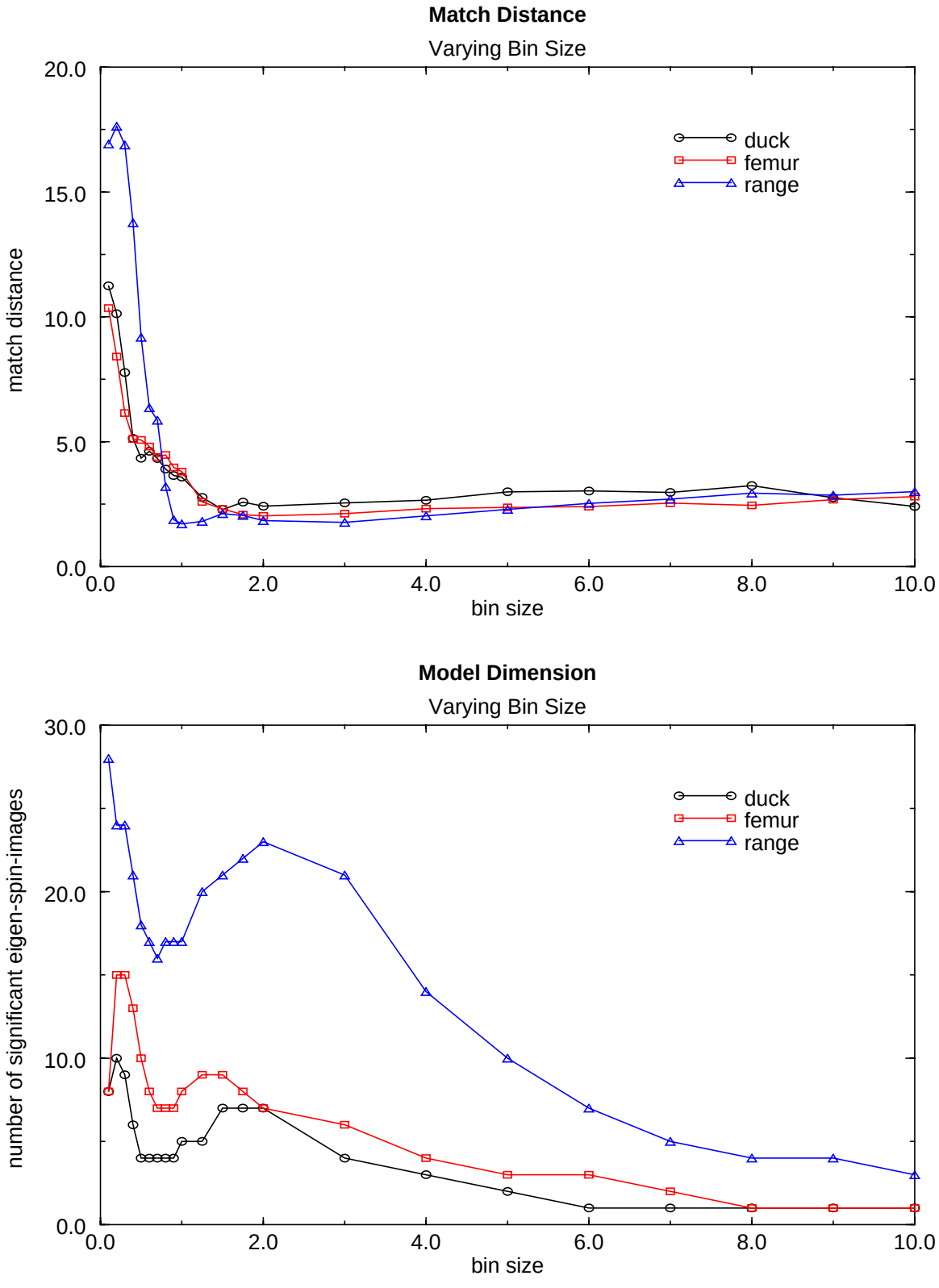


Figure 9-4: Effect of spin-image bin size on the match distance statistic and the model dimension while image width and support angle are fixed at 10 pixels and 180 degrees, respectively.

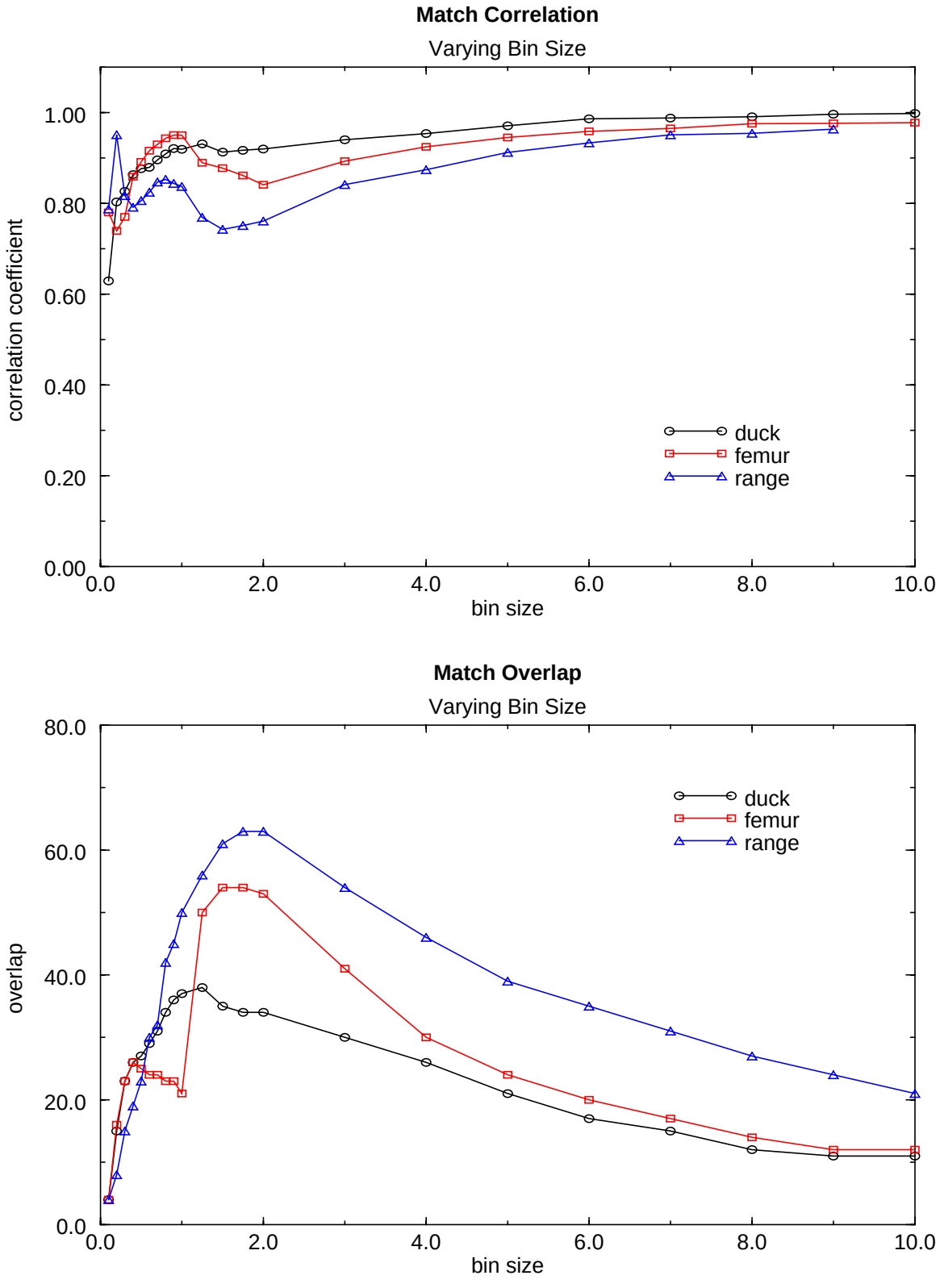


Figure 9-5: Effect of bin size on the match correlation and match overlap statistics as bin size is increased, while image width and support angle are fixed at 10 pixels and 180 degrees, respectively.

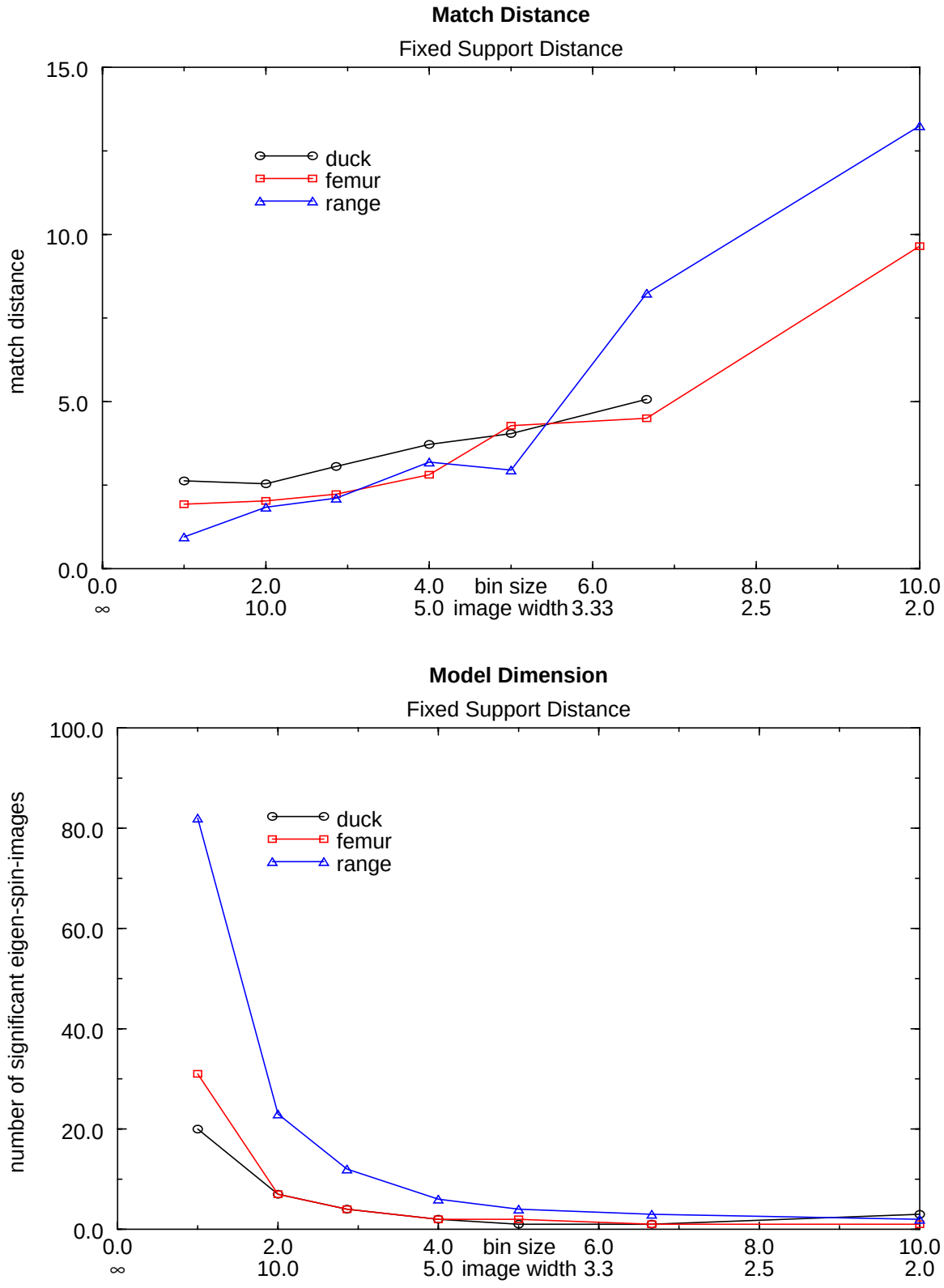


Figure 9-6: Match distance and model dimension for spin-images with constant support distance of 20.0, but with varying bin size and image width.

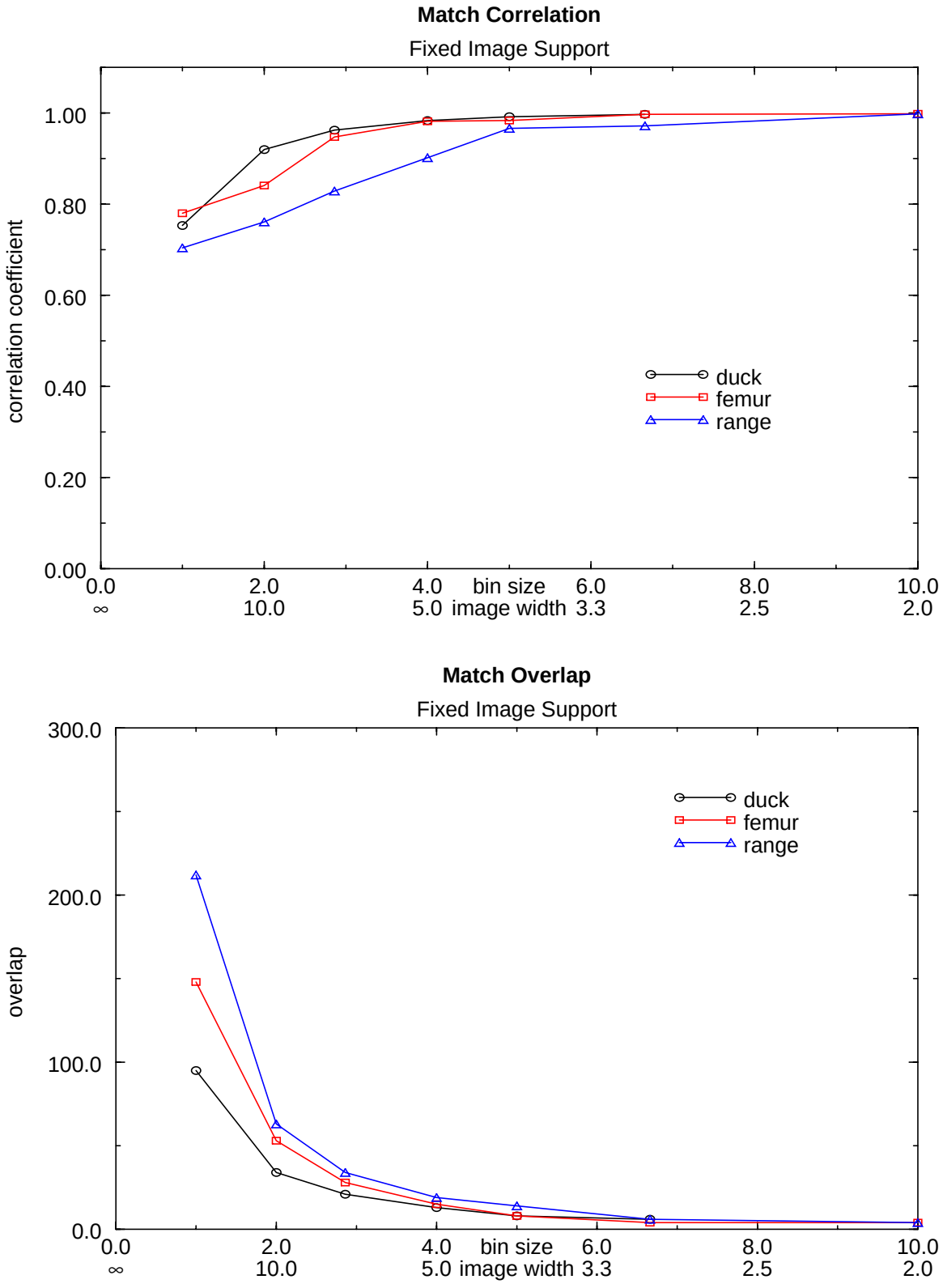


Figure 9-7: Match correlation and match overlap statistics for spin-images with constant support distance of 20.0, but with varying bin size and image width

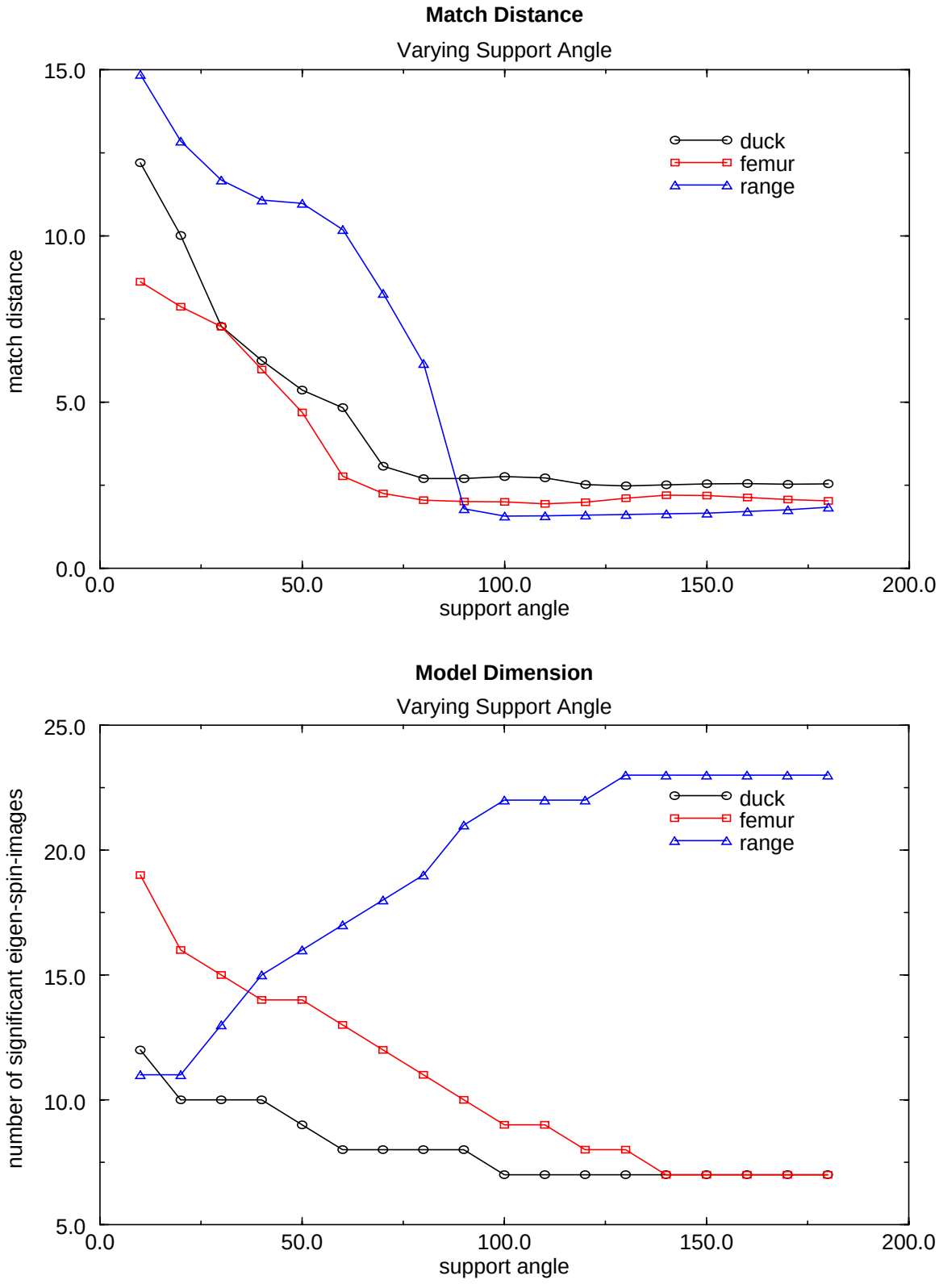


Figure 9-8: Match distance and model dimension for spin-images with varying support angle.

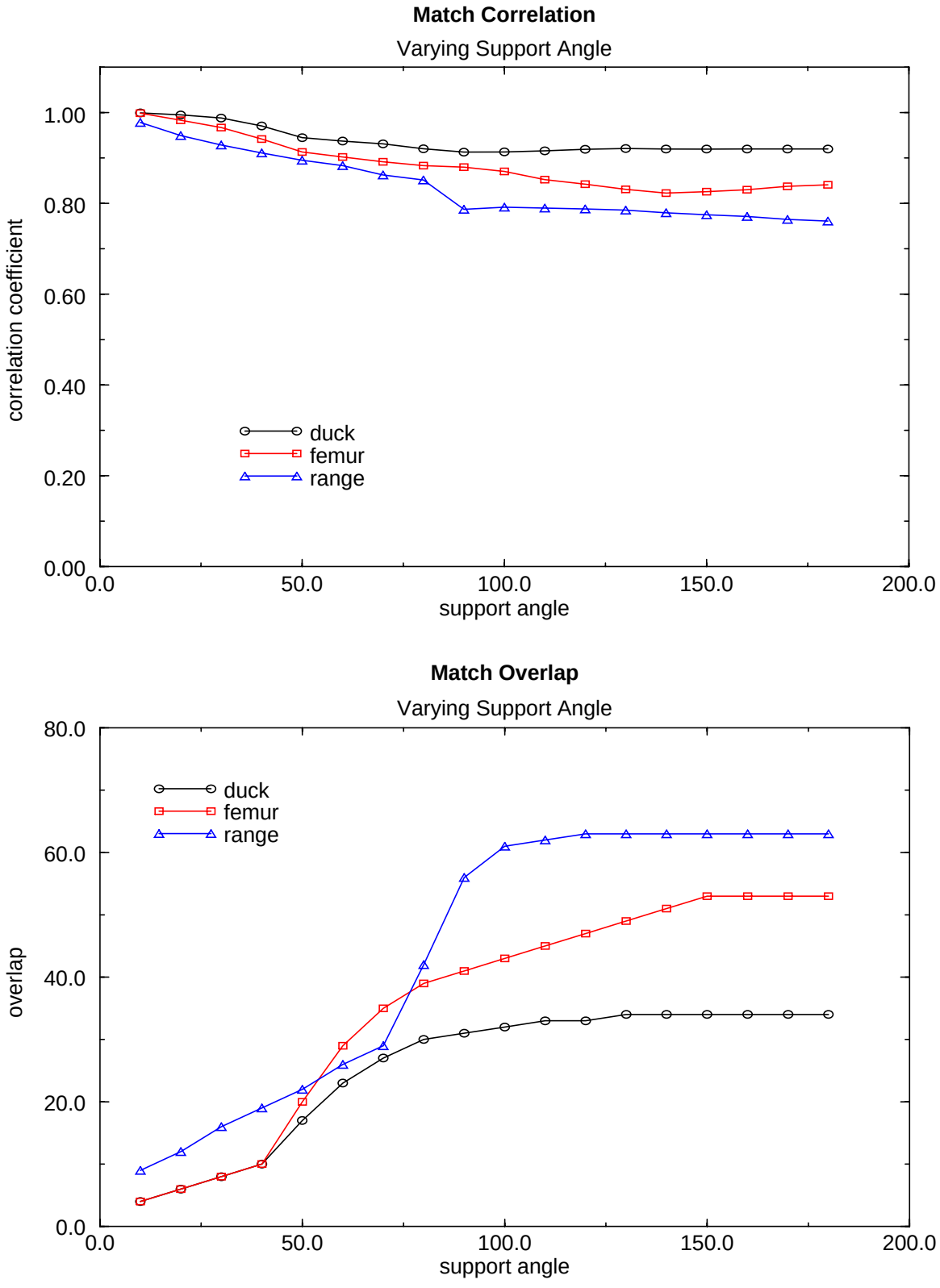


Figure 9-9: Match correlation and match overlap statistics for spin-images with varying support angle.

9.2 Matching Variables

Matching variables are differences between a model and a scene that can be quantified but, in general, are not adjusted by the user during surface matching. The three matching variables we investigate are the distribution of scene points, scene mesh resolution and scene noise. The following sections describe the metrics used to analyze the effect of matching variables on spin-image matching and then give an analysis of the three matching variables.

9.2.1 Matching Metrics

Analysis of matching variables occurs between two registered models: the control or base model and the model in which one of the matching variables has been changed. For example mesh resolution will be investigated by calculating matching statistics between the base model shown in Figure 9-1 and a model with the same shape and position, but a different mesh resolution. For purposes of discussion, the base model will be called the model; the model it is being compared to will be called the scene.

The first matching metric is the match distance as defined in Section 9.1.1. The match distance measures the median distance between corresponding points in the model and scene established though spin-image matching. Since the model and scene are assumed to be registered, the match distance statistic will be small when correct matches are generated.

The second metric is the number of significant eigen-spin-images in the scene. This metric, called scene dimension, is computed for a model using the technique described in Section 6.2 for a reconstruction level of 95%. The model dimension is the number of significant eigen-spin-images in the model. Differences between model and scene dimension indicate differences between the model and scene.

A novel metric for analyzing matching variables is the *shape correlation statistic*. The shape correlation statistic is calculated as follows. Select a vertex in the model and compute its spin-image by accumulating points in the model. Next, compute a spin-image using the model vertex; but instead of accumulating points in the model, build the spin-image by accumulating points in the scene. Since the model and scene are registered, the correlation between the two spin images will be large when the model shape is similar to the scene shape. Create and cor-

relate the above spin-images for all vertices in the model. The median of the computed correlation coefficients is the shape correlation statistic. Shape correlation will be close to one when the model and scene are similar in shape as measured by spin-images. The shape correlation statistic is a way to measure shape similarity among models when the models are assumed to be registered. Chapter 10 discusses in detail shape similarity metrics using spin-images.

The final metric is *subspace correlation*. Subspace correlation between a model and scene is computed as follows. Determine the significant eigen spin-images of the model and scene and order them from largest to smallest, based on eigenvalue. Set *D* to the maximum of the number of eigen-spin-images in the model and the number of eigen-spin-images in the scene. Create *D* pairs of eigen-spin-images by associating model eigen-spin-images with scene eigen-spin-images of the same order. The average correlation coefficient of all pairs of eigen-spin-images is the subspace correlation. When the significant eigen-spin-images in the model and scene are similar, the subspace correlation will be close to one. Subspace correlation is a method for measuring shape similarity between complete models without requiring registration.

The following sections describe the effect of matching variables using the four matching metrics. For each matching variable, different scenes are created that are related to the base models shown in Figure 9-1. The base models are resampled to 1000 points and scaled so that their mesh resolution is 1.0.

9.2.2 Surface Sampling

The first matching variable investigated is surface sampling. Surface sampling describes the position of the vertices of the surface mesh representing the shape of an object. Surface sampling is different from mesh resolution which describes the median distance between samples taken from the surface. In realistic situations, the scene surface sampling will be different from the model surface sampling, so spin-image matching must work in spite of different samplings. By setting bin size appropriately, the effect of surface sampling on spin-image matching can be made negligible. To verify this hypothesis experimentally, we created the following experiment. First, for each analysis model, we created two surface meshes with different surface samplings as follows. Using the duck model as an example, we took the model of highest resolution in Figure A-12 and added gaussian noise with a sigma of 0.1 times its resolution to each

of the 3-D coordinates of its vertices. We then decimated this fine mesh to a mesh containing 1000 vertices, giving us one of the surface meshes. Next, we applied gaussian noise to the vertices of the fine duck mesh, using a different random number seed. We decimated this mesh to 1000 points to obtain the second mesh. As shown in Figure 9-10, the two meshes have different surface samplings, but are of the same resolution and have an even distribution of points over their surfaces. The two meshes were scaled so that the mesh resolution was 1.0. Next, we computed the four matching metrics between the two surface meshes as bin size varied from 0.25 times the mesh resolution to 10.0 times the mesh resolution, while the image width was fixed at 10 and the support angle was set to 180 degrees. Figure 9-13 and Figure 9-12 show the effect of bin size on spin-image matching when the surface sampling between meshes is different.

Surface sampling has little effect when bin size is greater than mesh resolution.

Match distance is large when bin size is small, so incorrect matches are occurring. As bin size increases, match distance levels off around the mesh resolution, so correct matches are being made. When the bin size is below the mesh resolution, spin-images are uncorrelated because averaging by accumulating many points into each bin is not occurring during spin-image generation. The discrete position of vertices is controlling the appearance of spin-images. This is validated by the plot of shape correlation. Below the mesh resolution, shape correlation is low, so spin-images generated for scene points and models points are dissimilar. However, as bin size increases, the shape correlation approaches unity.

Below the mesh resolution, the subspace correlation is small, but as the bin size approaches mesh resolution, the subspace correlation approaches unity. This indicates that, taken as a

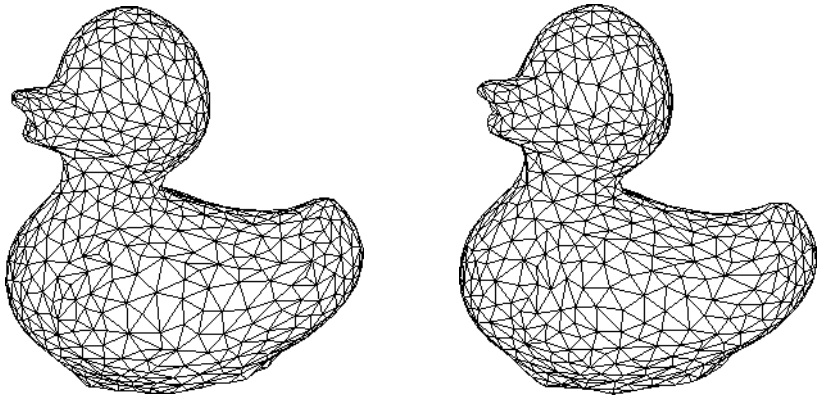


Figure 9-10: Duck meshes used to test effect of surface sampling.

whole, the spin-images in the model are becoming similar to the spin-images in the scene as bin size increases. The scene dimension is high but decreasing below the mesh resolution, then increases and finally decreases as bin size gets larger. Since this experiment is similar to the one performed for the bin size generation parameter, the variation of the scene dimension can be explained by the same forces that affected model dimension as detailed in Section 9.1.3.

9.2.3 Mesh Resolution

Mesh resolution is the median distance between vertices in a surface mesh. As mesh resolution increases linearly, the number of points in the mesh grows quadratically. Mesh resolution controls the number of points that fall into each bin in a spin-image. A mesh of low resolution will have a small number of points falling into each bin, so the accumulation in each bin, and hence scale of the spin-image will be small. A mesh of high resolution will cause many points to fall into each bin, so the scale of a spin-image will be high. Intuitively, spin-image matching can occur between meshes of different resolution because the major difference in spin-images will be due to scale and the correlation coefficient normalizes scale between images when comparing them. The following experiment verifies this fact. We generated a hierarchy of meshes with numbers of points ranging from 20 to 10000 for each of the analysis models using the algorithm described in Appendix A. The spin-images from the base model, each containing 1000 points, were then compared to the spin-images of the hierarchy of models with fixed bin size of 2.0, image width of 10 and support angle of 180 degrees. The matching metrics generated are shown in Figure 9-14 and Figure 9-15.

Spin image matching is insensitive to mesh resolution.

The matching distance between the model and scene was high for very low resolutions. Then, at around 300 points (one third the number of points in the base model), the matching distance leveled off around the mesh resolution. The matching distance stays low even when the base model is compared to meshes of much higher resolution (10000 points). Therefore, correct matches using spin-images can be obtained for a wide window of mesh resolution around the base model resolution; matching is insensitive to mesh resolution. The shape correlation statistic shows that, for low mesh resolutions, spin-images generated from points in the model and points in the scene are uncorrelated. This is because the shape of the model cannot be described by a small number of points and averaging in spin-image bins is not occurring. Therefore, the

spin-images generated from the base model and the scene will be different. At high resolution, the shape correlation statistic remains near unity, justifying the low match distances.

The scene dimension remains fairly constant as resolution increases because image width and support distance are fixed. Below 300 points, the scene dimension increases because averaging during accumulation is not occurring. Therefore, the spin-images will have more variation and hence more significant dimensions. The subspace correlation metric approaches unity as the base mesh resolution is approached, and it decreases as the mesh resolution changes from the base resolution. As mesh resolution increases, more surface detail appears in the mesh. This surface detail is represented in significant eigen-spin-images, causing the eigen-spin-images of the base model and higher resolution models to be different. Similarly, at low resolutions, surface details are lost and the vertices on the surface become more spread out. Spin-images all over the model become less correlated, making the principal directions of the spin-images ill-defined. This causes differences between the significant eigen-spin-images of the base model and the significant eigen-spin-images of the low resolution model.

9.2.4 Scene Noise

Scene noise is caused by sensing errors. Since spin-images encode global shape, and the value in each bin is the result of averaging the position of many points, matching of spin-images will be robust to significant levels of scene noise. To test this hypothesis, we created the following experiment. For each base model, a sequence of increasingly corrupted models was created by adding gaussian noise of increasing sigma to the 3-D position of the vertices in the mesh. The connectivity of the meshes remained the same. Adding noise to the vertex positions will corrupt the surface normals as well as surface position, because the normals are computed by fitting a plane to corrupted vertices. Examples of the some corrupted surface meshes for the duck model and the corresponding noise sigmas are shown in Figure 9-11. Sensing errors are generally systematic, so gaussian noise is a very simplified model scene noise. However, it is sufficient for demonstrating the effects of noise on spin-image matching. The matching metrics between the base models and the meshes of increasing corruption are shown in Figure 9-17 and Figure 9-16. In these experiments, spin-image bin size was set to 2.0, image width was set to 10 and support angle was set to 180 degrees. Since the base mesh resolution is scaled to unity, a noise sigma of 1.0 corresponds to point position errors that are on order of the resolution of

the mesh. In practical sensing situations, noise levels will be a small fraction of the mesh resolution. The normals used to compute oriented points are determined by fitting planes to the corrupted vertices in the mesh. Consequently, both normals and 3-D positions are corrupted.

Spin-image matching degrades smoothly as scene noise increases.

The match distance statistic starts out close to the mesh resolution and gradually increases as the meshes become more corrupted. No significant increase occurs in the match distance until the noise sigma reaches 0.5. A noise sigma results in significant corruption of the scene data (as demonstrated in Figure 9-11), so correct matching can occur even when the scene data is significantly corrupted. The slow increase in match distance demonstrates that the performance of matching gradually degrades as noise is added to the scene. In other words, there is no noise level at which the matching fails catastrophically.

Shape correlation of the models remains near unity at small noise sigmas and then drops off gradually. In our noise experiment, the complete effect of the scene noise is not accurately conveyed by shape correlation; the errors in surface normal that will corrupt the oriented point bases in the scene will not show up in the shape correlation metric. If the role of model and scene were reversed in the computation of shape correlation, then the shape correlation would be a more accurate metric for spin-image matching.

Scene dimension increases as noise increases because noise adds surface detail to the models. After the noise sigma reaches 1.0, it begins to decrease, because surface noise is making the shape of the models random. The reduction in shape information makes the spin-images more correlated, hence decreasing the scene dimension. Subspace correlation starts out near unity and rapidly decreases around noise sigma of 0.25. Since the scene subspace rapidly changes from the model subspace as noise is added, matching spin-images using compression is more sensitive to scene noise than matching without compression.

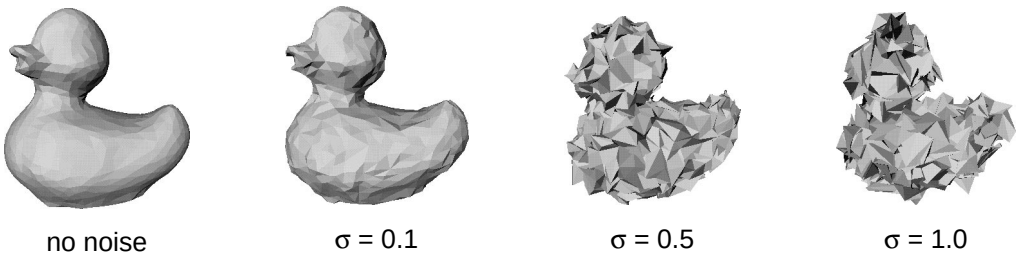


Figure 9-11: Shaded meshes of increasing corruption used in the analysis of scene noise.

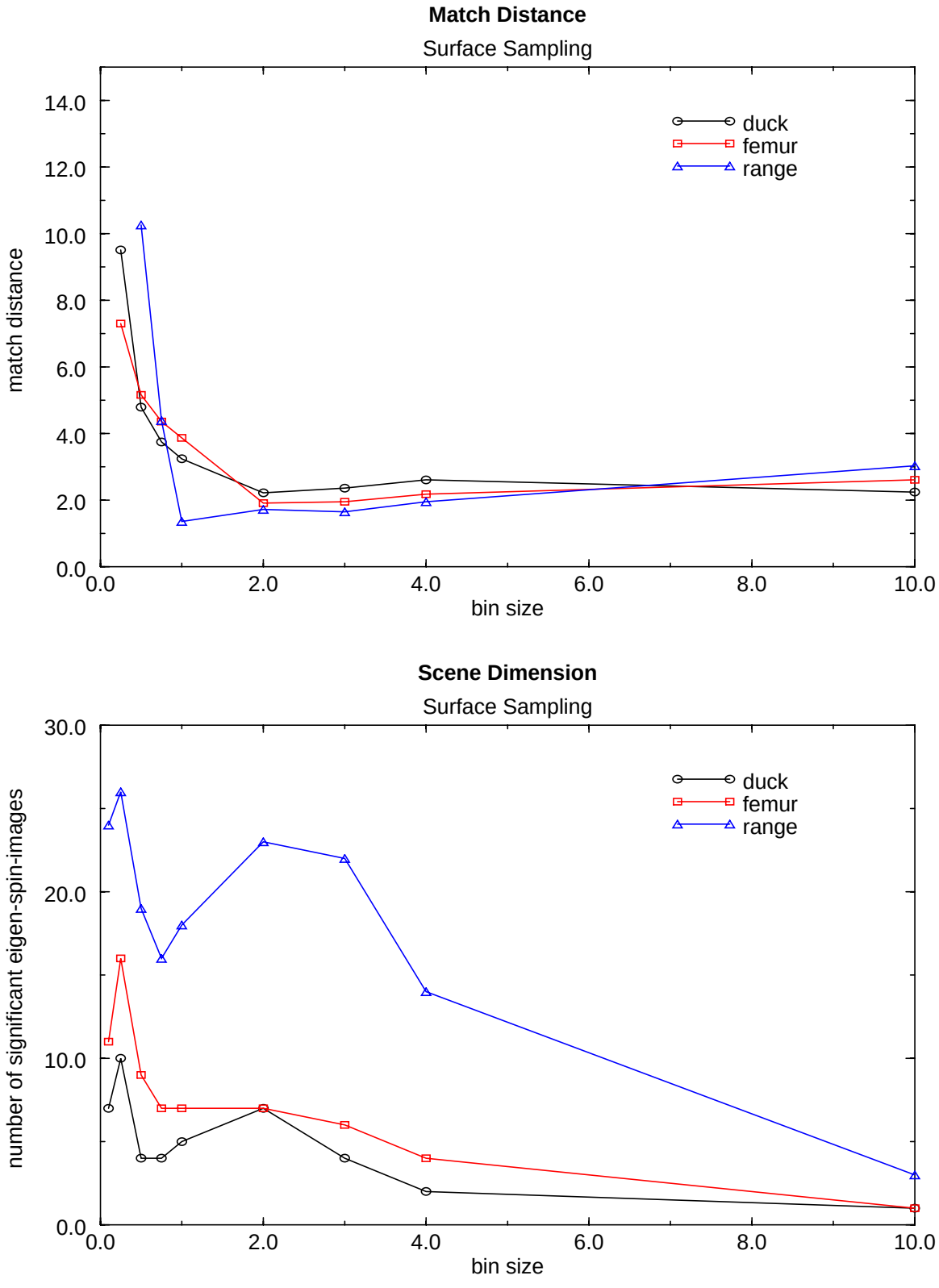


Figure 9-12: Match distance and scene dimension for determination of the effect of surface sampling on spin-image matching.

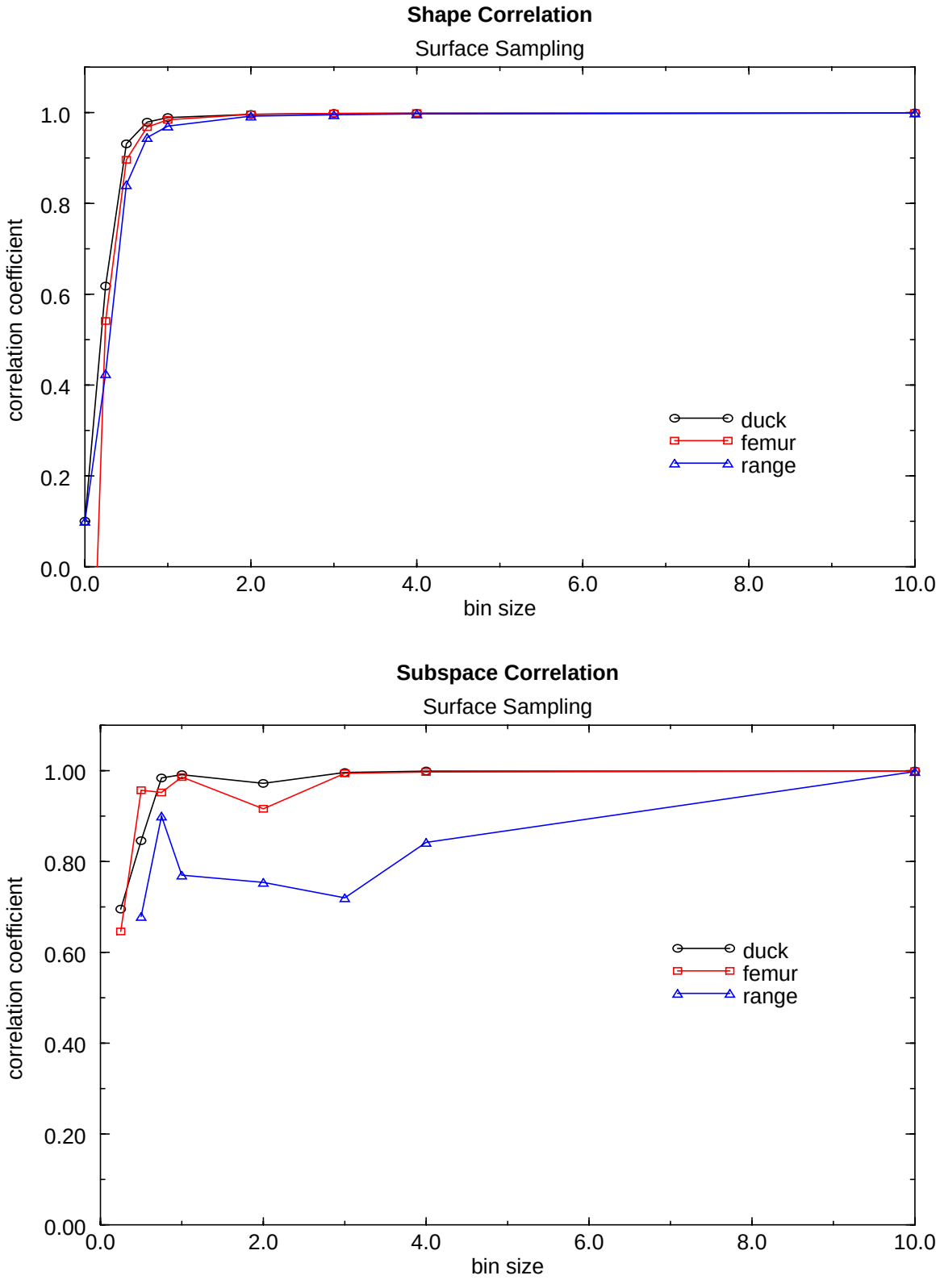


Figure 9-13: Shape Correlation and subspace correlation for determination of the effect of surface sampling on spin-image matching.

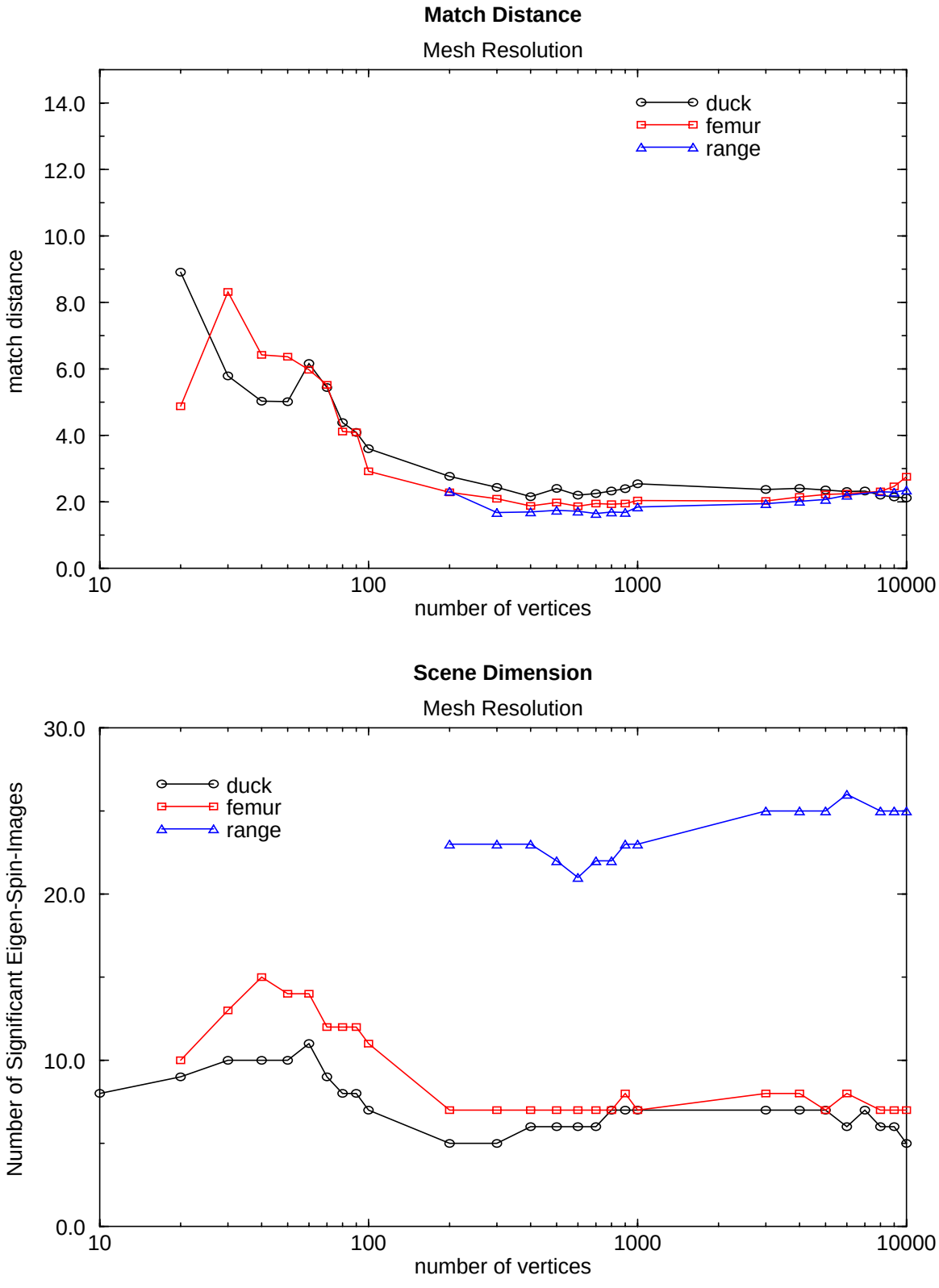


Figure 9-14: Match distance and scene dimension for determination of the effect of mesh resolution on spin-image matching.

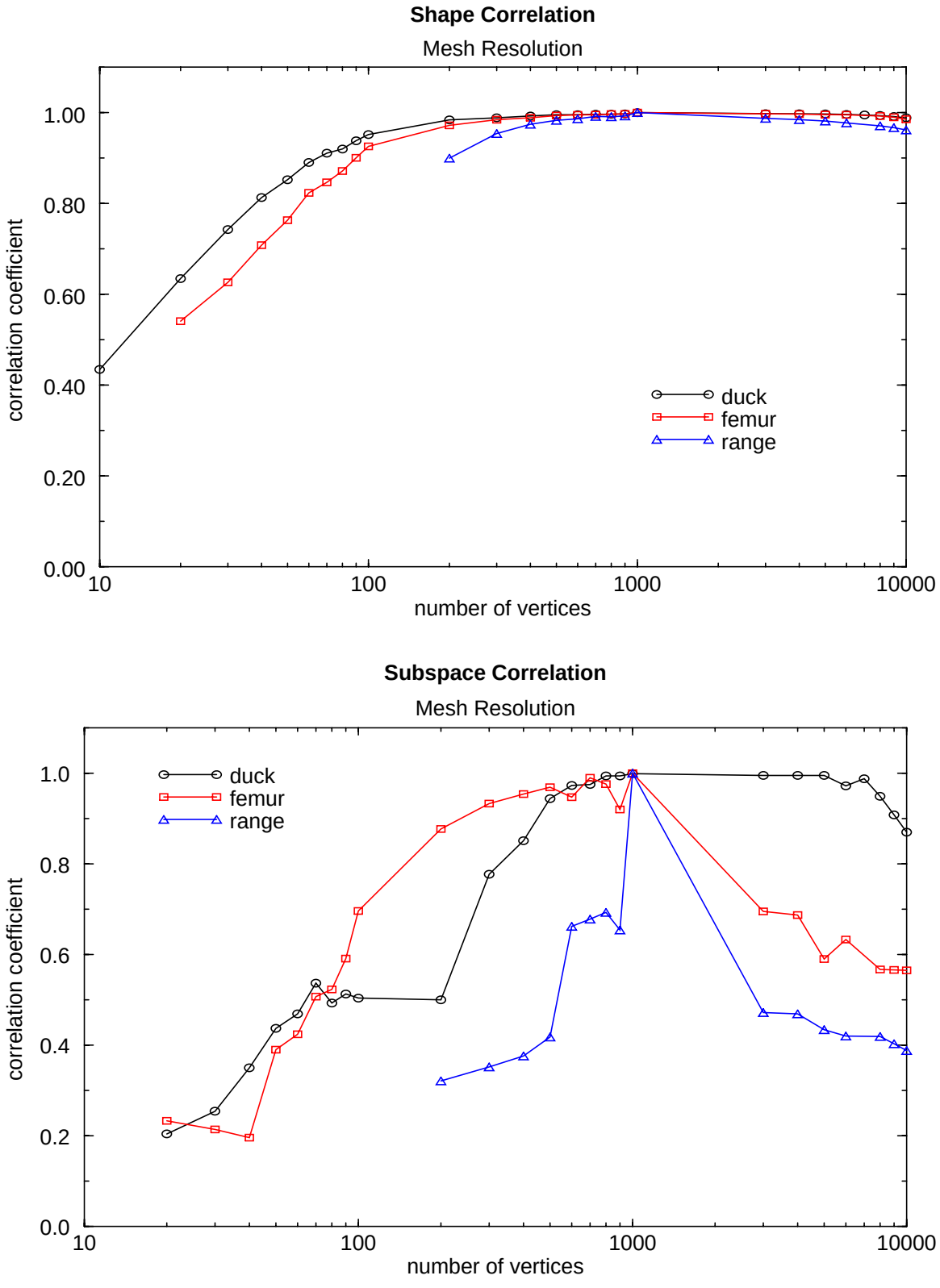


Figure 9-15: Shape Correlation and subspace correlation for determination of the effect of mesh resolution on spin-image matching.

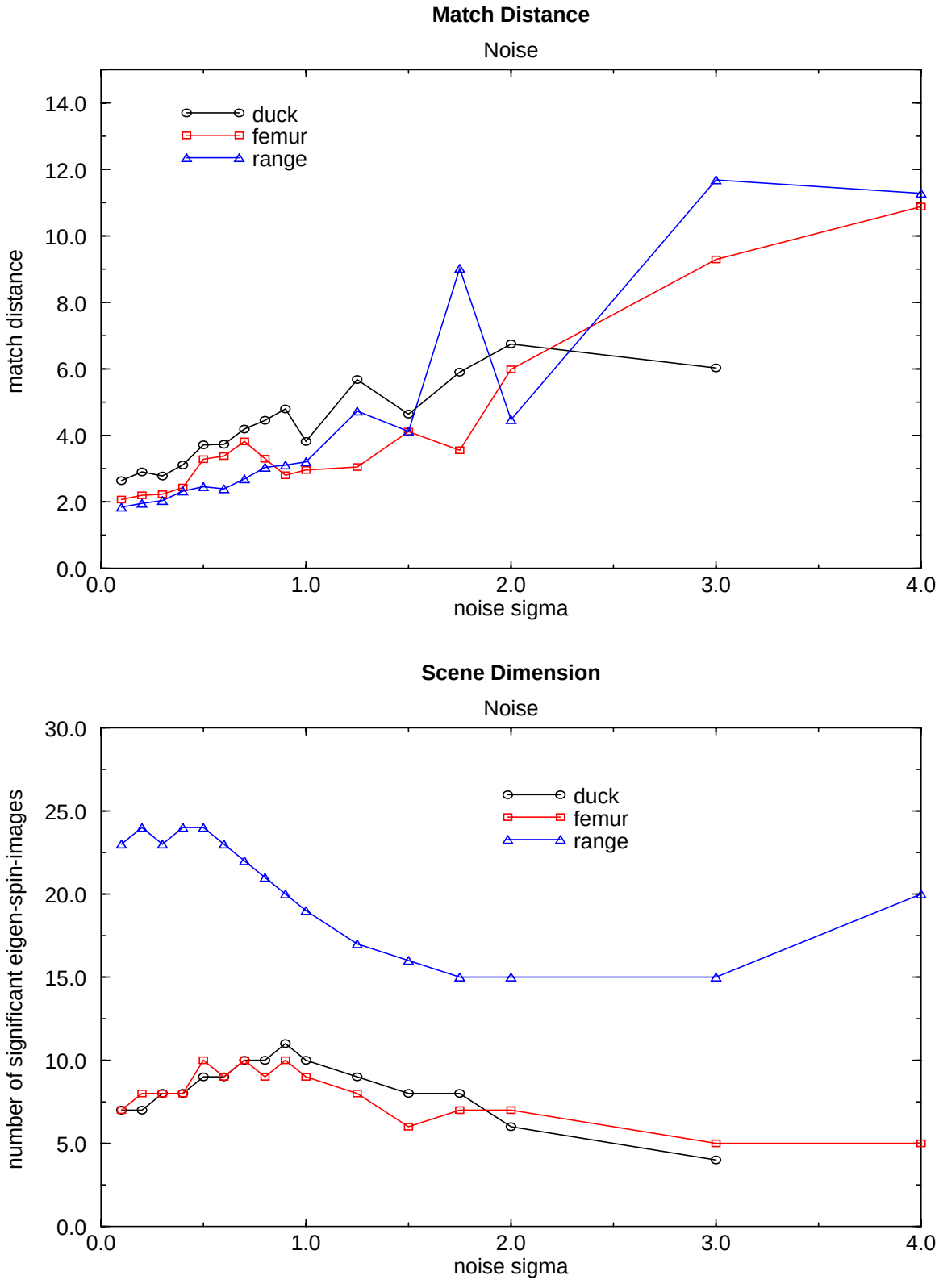


Figure 9-16: Match distance and scene dimension for determination of the effect of scene noise on spin-image matching.

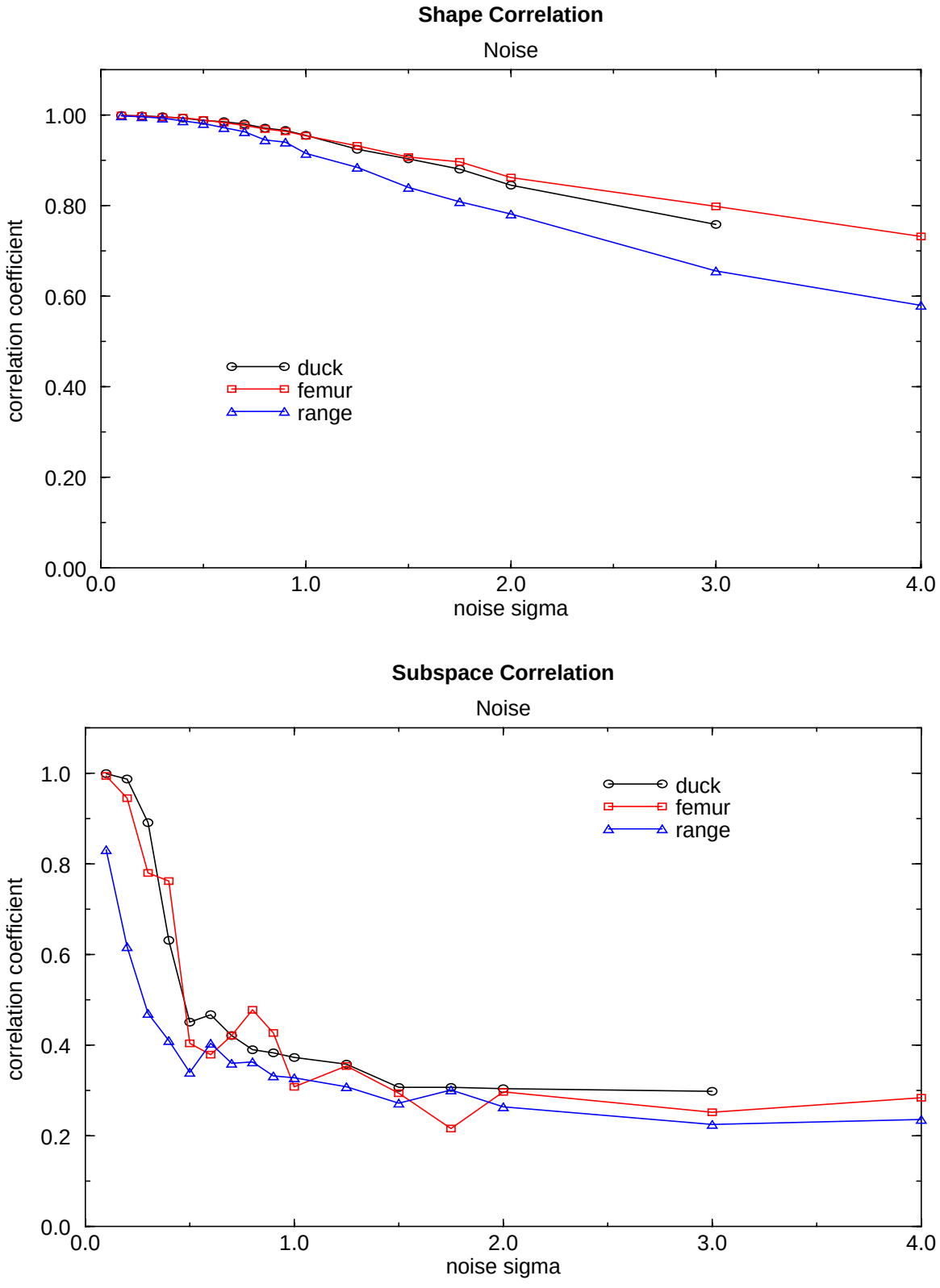


Figure 9-17: Shape Correlation and subspace correlation for determination of the effect of scene noise on spin-image matching.

Chapter 10

Future Work

In this thesis we have examined the spin-image representation in detail and have shown how it can be used for surface matching with applications in object recognition and surface registration. Spin-images are a general shape representation, so the applicability of spin-images to problems in 3-D computer vision is broad. Consequently, this thesis has not explored all possible applications and uses of spin-images. The purpose of this chapter is to describe possible future applications of spin-images in an attempt to demonstrate the generality our representation.

10.1 Shape Analysis

Shape analysis is the part of computer vision concerned with quantifying similarities and differences between the shape of objects. It encompasses shape similarity, shape symmetry, shape synthesis, part decomposition and object classification. Since spin-images can represent surfaces of almost any shape, they are a useful representation for shape analysis.

10.1.1 Shape Similarity

A shape similarity measure defines quantitatively the similarity in shape between two object surfaces. To date there is no well established shape similarity metric because defining differences between surfaces depends on the representation used to describe them. For example, with parametric representations a shape similarity metric can be defined as the difference between corresponding surface parameters. For the Spherical Attribute Image [38] (SAI) representation, shape similarity is defined as the sum of differences between curvature values of two registered

SAI [99]. These measures restrict the class of object that can be compared and require alignment of the surfaces before comparison.

Shape similarity is important because it can be used for object classification; if two objects are similar then they can be classified in the same group [99]. Classification can be exploited to make object recognition more efficient. By first recognizing the classes of objects in the scene and then the actual object instances in the classes, the number of comparisons to models in the model library is reduced. Shape similarity, object classification and hierarchical object recognition are tightly coupled in 3-D computer vision.

In Chapter 9, we developed some measures of shape similarity based on spin-images. Of the measures, the best measure of shape similarity is the subspace correlation metric (Section 9.2.1) because it can compare surfaces without alignment. The subspace correlation metric measures the average correlation coefficient between the significant eigen-spin-images describing two surfaces. In Chapter 9 this metric was used to measure the similarity between surface with different surface sampling and resolutions. This metric can also be used to compare objects of different shape. In Figure 10-1, the subspace correlation between the rabbit surface (on the left) and four other surfaces is shown below the surfaces. As the surfaces warp from the rabbit shape to the bunny shape, the subspace correlation decreases. The subspace correlation of the rabbit surface with itself is 1.0. Using eigen-spin-images to measure shape similarity has the advantage that the surfaces do not have to be aligned before their shapes are compared. Fur-

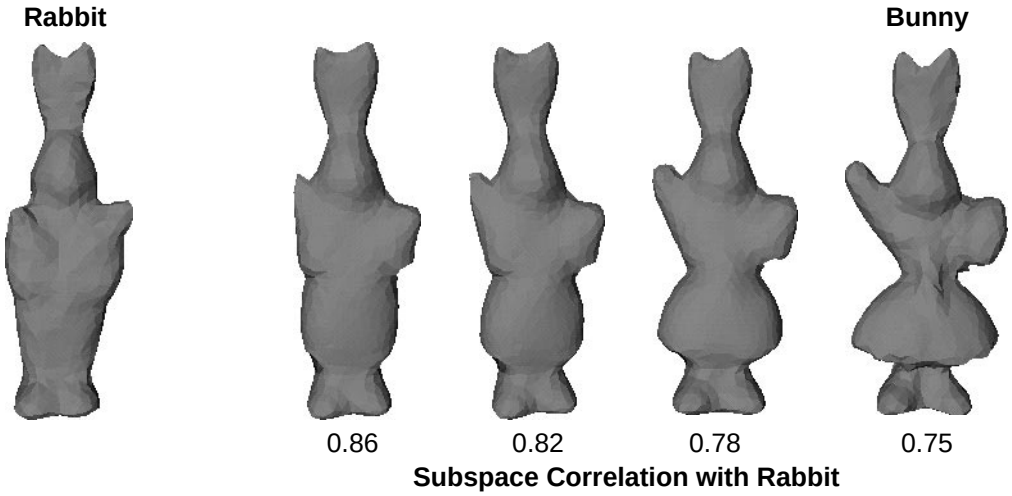


Figure 10-1: Shape Similarity. As rabbit shape warps into bunny shape, the subspace correlation metric indicates that the shapes become less similar.

thermore, when spin-images are generated with large support, eigen-spin-images create a basis in which to describe the global shape of the object; eigen-spin-images of large eigenvalue describe overall shape while eigen-spin-images of small eigenvalue describe smaller details. Using a systematic approach, a relationship between eigen-spin-images and surface shape could be established. This would give insight into ways to represent surface shape and possibly create new methods for object classification based on shape.

There exist similarities between eigen-spin-images and the modal representations of Pentland and Sclaroff [68][77]. Modal representations are global representations of objects created using finite element methods that describe the deformable modes of a complete object. By matching deformation energies, correspondences between objects can be established. By drawing analogies between modal representations and eigen-spin-images, it may be possible to use spin-images for deformable matching and comparison of deformable shapes.

10.1.2 Shape Synthesis

The ability to invert the spin-image generation process in order to synthesize surface shape is a useful capability. In other words, we would like to be able to take a set of spin-images and generate the surface corresponding to them. This is a tremendous optimization problem which involves solving for all of the positions of the oriented points used to create the spin-images based on comparison of spin-images. Solving this may not be practical. However, simplifying assumptions may make the problem solvable and still useful.

If a shape synthesis procedure were available, then the contribution of individual eigen-spin-images to surface shape could be determined as follows. Suppose the spin-images and eigen-spin-images for a surface have been computed. Replace the surface spin-images with their projection into the subspace spanned by the top eigen-spin-images. Using the projected spin-images, synthesize the shape of the new surface. The new surface will contain spin-images lying completely in the subspace spanned by the top eigen-spin-images, so the surface will describe the contributions of the top eigen-spin-images to surface shape.

To solve this shape synthesis problem, one could imagine an iterative optimizations procedure that adjusts the position of the oriented points on the surface to positions that are more consistent with those described by the projected spin-images. In other words, the original surface

could be used as a initial condition for an iterative shape synthesis algorithm.

10.1.3 Part Decomposition

Another advantage of spin-images is their ability to scale from local to global representation. Spin-images with large support convey the global shape of an object while spin-images of small support convey the shape of objects in a local neighborhood around the oriented point. Spin-images are localized to make matching robust to clutter and occlusion, but localization can also be used to facilitate local shape matching. Consider the bunny and rabbit models shown in Figure 10-1. The ears, feet and heads of the models are similar while the bodies of the models are not. If the surfaces of the models are represented with local spin-images, then the spin-images from the ears, head and feet of the models will be similar while the spin-images from the bodies will be different. Therefore, local parts of the surface can be matched even when other parts of the surface are different.

The ability to match local parts of surfaces hints at a method for parts decomposition or segmentation of surfaces. By grouping surface patches from different models into the same class (e.g., ears, head, feet, body), model surfaces can be represented by a set of parts instead of a set of oriented points. This grouping will make matching more efficient, because a model library will be representable by a description of all parts in the library along with part decomposition for each model in the library. Recognition will proceed by searching for parts in the scene and then matching recognized parts to model part decompositions. This form of recognition is similar to the recognition work of Dorai and Jain [21] and Arman and Aggarwal [1] where objects are recognized by matching graphs of surface patches.

10.1.4 Shape Symmetry

Spin-images can also be used to detect object symmetries. As shown in Figure 6-2, when an object is symmetric, spin-images on opposite sides of the plane of symmetry will be similar. In contrast to similarity between spin-images from adjacent mesh vertices, corresponding points on opposite sides of a plane of symmetry have similar spin-images, but dissimilar 3-D coordinates. This observation can be exploited to detect object symmetries.

Because points on opposite sides of a plane of symmetry have similar spin-images, their pro-

jection onto eigen-spin-images will be similar. Consider a 2-D example using the model of a elbow joint shown in Figure 10-2. The elbow has a plane of symmetry parallel to the *x*-*z* plane. The 2-D spatial contour made by intersecting the elbow surface and the plane $z = z0$ clearly shows this symmetry. Each point along the spatial contour has an associated spin-image. As in spin-image compression, these spin-images along the contour can be projected onto the top *s* eigen-spin-images for the pipe surface. This will result in a set of *s*-tuples for points along the contour. A plot of the first three coordinates of these *s*-tuples with respect to the *x* coordinate of the spatial contour are shown in Figure 10-2. These spatial/spin-image contours all repeat themselves; the contour is traversed for one side of the plane of symmetry and then repeated for the other side. Positions in the contour that are far apart based on arc length are close in *s*-tuple coordinate.

This example sets up discussion of the general problem. If the coordinates of vertices in the surface mesh are replaced by the s-tuples of spin-image projection coefficients, a 2-D manifold in s-dimensional space will be created. This manifold is similar to the manifold presented by Murase and Nayar [65] for appearance-based recognition except that the 2-D parameters are

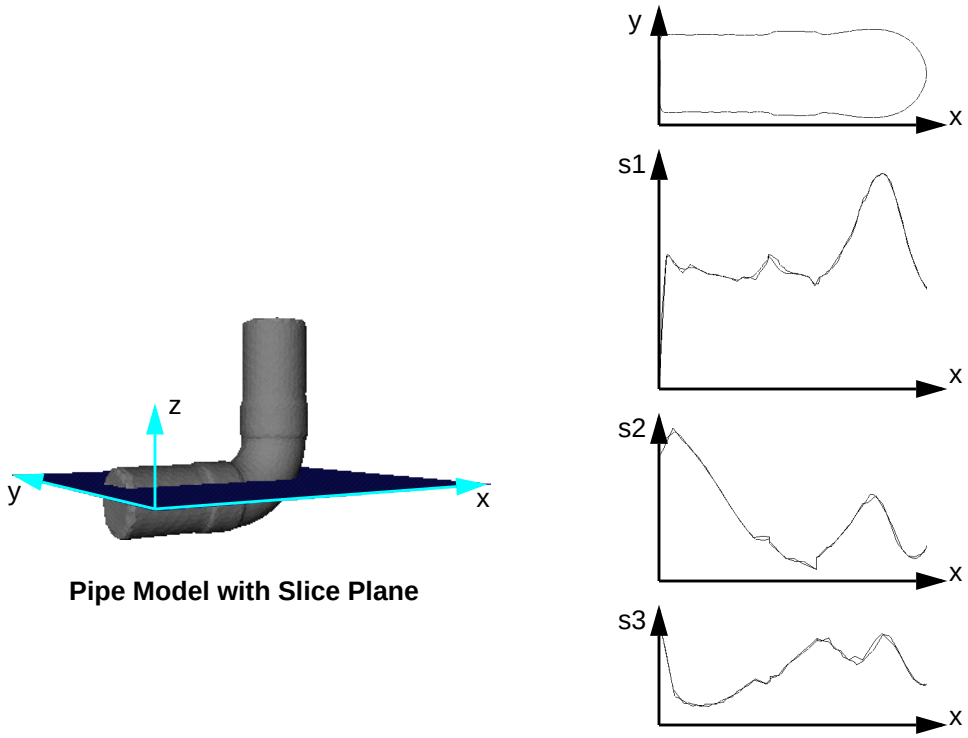


Figure 10-2: Shape symmetry. The elbow joint contains a plane of symmetry which is indicated by the 2-D spatial contour created by intersecting a plane with the surface. This symmetry is represented in spin-images as a repetition of spin-image projection coefficients along the contour.

surface position instead of object pose and illumination direction. For symmetric surfaces, this manifold will overlap or repeat itself in s-D space. By detecting repetitions of the manifold, object symmetries can be detected. Furthermore, the number of planes of symmetry will be indicated by the number of repetitions of the manifold shape.

Object symmetry presents another avenue for making recognition more efficient. If an object contains a plane of symmetry then only half of its spin-images are unique. By comparing scene spin-images to the unique half of spin-images in the model, the spin-image matching time is cut in half. Of course, when a scene spin-image is matched to the model, correspondences between the scene oriented point and both oriented points on opposite sides of the model plane of symmetry must be created.

10.2 Extensions

Our initial implementation of surface matching demonstrates that spin-images are an effective representation for matching surfaces. However, the following algorithmic additions could be used to make spin-image matching more efficient and robust.

10.2.1 Learning

It is likely that some spin-images are more descriptive or robust to noise than other spin-images. These spin-images will be better candidates for matching because they will match fewer points with less matching errors. If these spin-images could be determined before surface matching, then spin-image matching could be made more robust by only matching spin-images that make consistently good matches. Furthermore, by comparing only the most descriptive and robust spin-images during matching, the number of correlations is reduced, making surface matching more efficient.

The best spin-images for matching a model to scene data can be learned if many surface matching experiments are performed. Each time a model is correctly matched to a scene, model spin-images that are used to match the surfaces are given a positive vote. Each time a model is incorrectly matched to a scene, the model spin-images used to match the surfaces are given a negative vote. After accumulating the votes for many recognition trials, the overall vote for each spin-image will determine its ability to correctly match a model to a scene.

If we know which spin-images are better for matching then possibly a different form of the matching engine is appropriate. For instance, it might be better to select a point from the model that is good at matching and then search for it in the scene. Once enough matches between model and scene are found, the model can be aligned with the scene and its position verified. This form of matching would be similar to the alignment form of recognition [46].

10.2.2 Coarse to Fine Recognition

We have a method for smoothly varying the resolution of surface meshes (described in Appendix A). In Chapter 2, we showed that spin-images of varying resolution can be made by varying bin-size during spin-image generation. Using these control of resolution tools, coarse-to-fine models for recognition can be generated as follows. First, create the spin-images for the finest resolution (most vertices) surface mesh for an object using a spin-image bin-size equal to the resolution of the surface mesh. Next, resample the surface mesh to decrease its resolution to one half the resolution of the original surface mesh. Create the spin-images for this model using a bin-size equal to half the resolution of the original surface. Repeat until the surface mesh resolution cannot be changed without drastically changing the shape of the object. The result is a pyramid of surface meshes and corresponding spin-images. At the top level of the pyramid is the coarsest surface mesh with the coarsest spin-images. At the bottom of the pyramid, is the

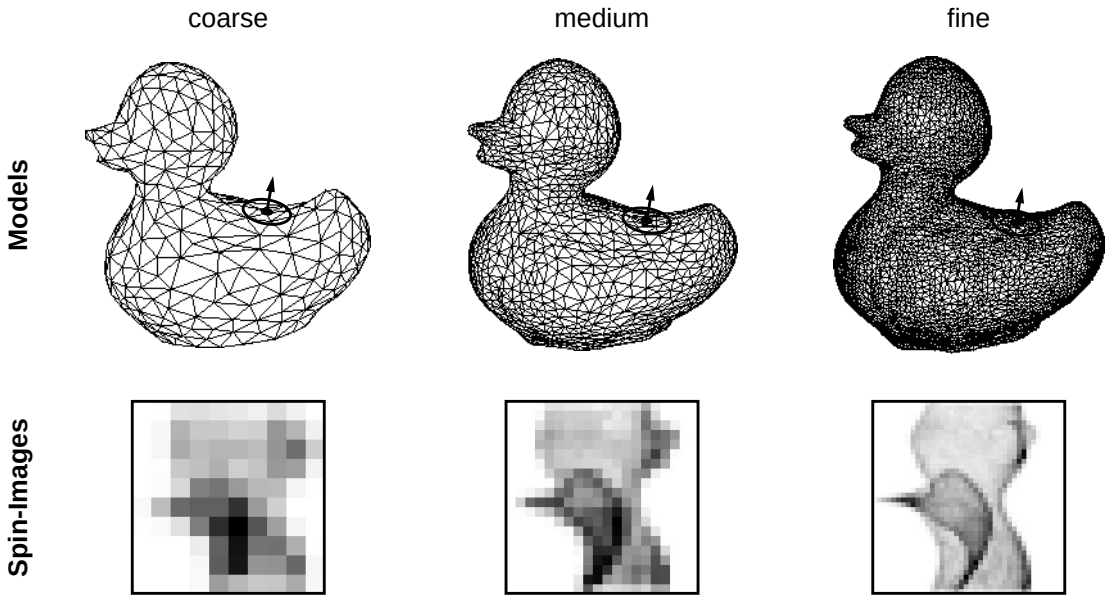


Figure 10-3: An example of a model pyramid. At the coarse level, the model mesh and spin-images contain less detail than the models at finer resolution.

finest surface mesh with spin-images containing the most detail. Figure 10-3 shows a three level model pyramid. Surface meshes and associated spin-images of increasing resolution are shown from approximately the same surface position.

Using this model pyramid, a coarse-to-fine recognition scheme, similar to the use of image pyramids in coarse to fine stereo matching, can be developed. First, a coarse mesh with coarse spin-images is matched to a scene. These coarse matches are then used to constrain the distance between matches at a finer level of resolution. By limiting the distance between corresponding points in the model and scene, the number of possible matches between model and scene spin-images is reduced. Consequently, the total number of spin-image comparisons is reduced. With coarse to fine resolution, is it conceivable that spin-image matching could be made logarithmic instead of linear in the number of points in a model.

10.2.3 Different Parameterizations

When generating spin-images, a cylindrical coordinate system is used to parameterize oriented point basis coordinates. It is possible that other coordinate systems may lead to more robust matching of spin-images. In particular, using a spherical coordinate system to describe oriented point coordinates may result in improved performance with respect to surface normal error. In this spherical coordinate system (shown in Figure 10-4) the two coordinates would be the distance ρ of the point to the oriented point vertex and the angle ϕ between the vector from the oriented point to the point and the tangent plane. In the cylindrical parameterization, the effect of surface normal error increases as the α and β increase. In the spherical coordinate system, the effect of surface normal error is constant for ρ and ϕ. By evenly distributing this error, spin-images generated with the spherical coordinate system may be matched more robustly. A consequence of the spherical coordinate system is that the volume swept out by bins of fixed ϕ in-

Figure 10-4: A comparison of two possible parameterizations for oriented point bases.

creases as ρ increases. This non-uniform bin-size may require modifications to spin-image generation and matching.

Support distance and support angle are thresholds used to localize spin-images to make them robust to clutter and occlusion. Implicit in the use of these thresholds is the assumption that clutter cannot occur within these thresholds. In reality this assumption does not hold in all cases; cluttering surfaces maybe inside or outside of the bounds set by these thresholds. Consequently, useful surface data for matching may be cropped out by the support thresholds. An alternative to the hard support thresholds is a support function, a function that assigns weights to spin-image bins based on their likelihood of corruption by clutter. Bins close to the oriented point are given large weight (since they are less likely to be corrupted by clutter) while bins far from the oriented point are given smaller weight. Spin-images are then compared by computing the weighted correlation coefficient between two spin-images, each bin being weighted based on its support function value. The smooth support function will result in more accurate matching because all (local and global) surface information is used, but it will also require larger spin-images which leads to an increase in storage size and matching times.

10.3 Applications

We have applied spin-image matching to surface registration and object recognition. Since spin-images describe the shape of a surface around a specific point they can also be used for 3-D surface tracking. Furthermore, through slight modification of spin-image creation, spin-images applicable to volumetric image registration can be created.

10.3.1 Tracking

Tracking is the problem of computing the position of an object as it moves over time while being imaged. Since the object position changes only a small amount between image frames, approximate knowledge of the object position is known in advance. Tracking usually proceeds by matching features in one frame to features in the next frame using the approximate transformation between frames to guide the matching process. When tracking 3-D surface data, spin-images can be used as features.

During spin-image matching, no knowledge of the transformation between surfaces is used.

However, if this knowledge is available, then it can definitely be incorporated to speed up spin-image matching. For example, an ICP algorithm can be devised that combines 3-D position and spin-images. A k-D tree that stores 3-D position and spin-image coordinates (probably better if the spin-image is compressed) could be created. Using this k-D tree closest points could be searched for based on proximity in 3-D position and spin-image coordinates. An appropriate balance between spin-image coordinates and 3-D positions would have to be found, but the end result would be a surface tracking algorithm based on spin-images. By adjusting the support of spin-images, tracking could be applied to both rigid and articulated bodies.

10.3.2 Volumetric Image Registration

In this thesis we have constructed spin-images from surface data. However, it is also possible to apply the concepts of spin-image matching to registration of volumetric images. Volumetric images are common in medical imaging where they are generated using various sensing modalities including Computed Tomography, Positron Emission Tomography and Magnetic Resonance Imaging. In all cases, the volumetric image is made up of voxels which contain an intensity that describes some property of the tissue being imaged. Volumetric image registration is important for comparison of data sets over time for surgical guidance and among patients for analysis of pathologies.

A slight variation in spin-image generation is needed to compute spin-images from volumetric images. Oriented points are created for each voxel in the image by substituting 3-D voxel position for surface position and substituting the gradient of the volumetric intensity for surface normal. No surface extraction is needed although less voxels will have to be matched if the surface voxels can be detected. The spin-image for each “oriented-voxel” is made by accumulating voxel intensity in the same way the that 3-D points are accumulated in surface based spin-images. The only difference being that the spin-image bin that a voxel maps into is incremented by the intensity of the voxel instead of just being incremented by one. Bilinear interpolation can be used to smear the contribution of the voxel in the image. After constructing spin-images in this way, voxels can be matched in the same way that oriented surface points are matched.

An advantage of using spin-images to match volumetric images is that no knowledge of the transformation between images is needed, and arbitrary rigid transformations can be handled.

Furthermore, since spin-images are compared with normalized correlation, differences in scale between image intensities can be handled. By localizing the support of voxel spin-images, local spin-images can be created which can be used to register images that have undergone deformations. Finally, Since spin-image matching is robust to clutter and occlusion, the effect of locally large differences between images (e.g., tumors, lesions) can be removed from image registration in same way that the effect of clutter is reduced in surface matching.

Chapter 11
Conclusion

We have presented a new data level representation for 3-D surface matching based on matching of spin-images generated using oriented points on the surface of an object. The effectiveness of our new representation comes from its ability to combine the descriptiveness of global shape properties with the view invariance of local features. We have demonstrated its effectiveness by showing results from simultaneous multi-model object recognition in cluttered scenes with occlusion and surface registration for object modeling.

11.1 Contributions

The major contribution of this thesis is the development of spin-images, a new representation for surface matching. Spin-images have many advantages that make them a significant contribution to the field of computer vision.

Spin-images are general.

We represent the surface of objects by polygonal meshes. By increasing the resolution of the mesh, objects of arbitrary detail and complexity can be represented. Spin-images are constructed from the polygonal mesh representation of objects, so they can be used to match the surfaces of objects with arbitrary shapes and topologies. We have shown surface matching results between algebraic, polyhedral and free-form surfaces, thus experimentally validating our claim.

Spin-images are also sensor independent. Surface meshes can be constructed from many different types of 3-D sensors. Surface meshes do not store information about the surface sensing process, so they are sensor independent representations. Since spin-images are constructed from surface meshes, they too are sensor independent. Because they are sensor independent,

spin-images can be used to match surfaces generated from different modalities. For instance, a complete model generated from CT can be matched to a partial view taken by a range sensor (Figure 3-10).

Spin-images are robust to scene clutter.

We have designed a robustness to clutter into spin-image generation and spin-image matching. By adjusting spin-image support parameters, spin-images that are localized close to the object surface are generated. By localizing spin-images, clutter is less likely to affect spin-image matching. This conclusion is born out by a theoretical model that explains why spin-images are resistant to clutter. The key to the model is that solid objects cannot intersect, so there exists a volume around an object's surface where clutter cannot exist. Furthermore, the similarity measure for comparison of spin-images masks some effects of clutter by comparing spin-images only in regions where both have data. Finally, the resistance of spin-images to clutter is demonstrated in statistics generated from hundreds of recognition experiments where complete objects are matched to cluttered scenes. The conclusions of these experiments is that, in practice, clutter does not affect spin-image matching if appropriate precautions are taken.

Spin-images can match complete surfaces to partial scenes.

By reducing the support angle of spin-images, spin-images generated from a complete object surface can be adjusted to resemble the spin-images from a scene containing the objects acquired from single view 3-D scanner. The support angle reduces the effect of self occlusion on matching. Matching complete object to partial scenes is demonstrated with hundreds of scenes. The conclusion from these recognition experiments is that spin-images can match surfaces with large amounts of occlusion, but the performance of matching degrades as occlusion increases.

Spin-images are simple to construct.

Spin-images are data-level representations that are created from a collection of 3-D points without surface fitting or optimization. This is in contrast to other surface representations for complex objects, such as generalized cylinders, super quadrics and deformable surfaces, which generally require a procedure to fit the surface model to 3-D data. Spin-images are simply generated by accumulating surface information in 2-D arrays.

An advantage of the simplicity of spin-images is that they are easy to analyze and understand.

Since spin-image generation is simple, determining the effects of variations in the 3-D data on spin-image matching and generation is straightforward. We have demonstrated the ease of analysis of spin-images in Chapter 9, where numerous experiments quantifying the performance of spin-images with respect to variables in the data and spin-image generation process have been analyzed.

Spin-images are widely applicable.

By modifying spin-image generation parameters, spin-images can be generated for many different applications. Spin-images can be used for accurate matching of complete object surfaces, for instance in the case of matching CT or MRI surfaces, by encoding the entire object shape in the spin-images. Spin-images can be used to match surfaces of different resolution as long as enough surface detail is present. This is especially relevant in the case of matching a large, coarse model to a dense, local model. Spin-images can match complete surfaces to partial surfaces and can do so in the presence of clutter and occlusion. Spin-images can also be used to register partial views of a surface, even when the surfaces have small amounts of overlap.

Spin-images are appropriate for object recognition.

We have demonstrated the use of spin-images for recognition of complex models from a model library containing twenty complete models. We have validated experimentally that spin-images perform well when recognizing objects in cluttered scenes with occluded models. To our knowledge, no one else has reported simultaneous recognition of multiple 3-D models in cluttered scenes with occlusion for a model library of up to 20 models on hundreds of scenes. Therefore, we have advanced the state of the art in computer vision by creating a recognition algorithm that works under more difficult situations than previously reported.

Spin-images work in practice.

We have demonstrated the effectiveness of spin-images in three domains. First we showed how spin-images can be used for registration of partial views of an object. This registration of partial views lead to a procedure for building models of complex objects without a calibrated apparatus for acquisition of range images. Successful object modeling was demonstrated by building twenty models of objects containing, polyhedral, algebraic and free-form surfaces. The second application was object recognition. We showed that object recognition works when many models are in the object library and in scenes that contain clutter and occlusion. Furthermore, by

recognizing free-form and parametric objects simultaneously, we demonstrated that spin-images can recognize a wide variety of objects present in the real world. The final application of spin-images was interior modeling. Using spin-images, a semi-automatic system that builds models of man-made interiors was demonstrated in live demos. This indicates that matching with spin-images is fast and robust enough to be effective in real situations.

11.2 Lessons Learned

Our object recognition system demonstrates that multiple models can be recognized simultaneously in scenes containing clutter and occlusion, without feature extraction or segmentation. From the beginning of this thesis work, we designed our representation and matching algorithms to work without feature extraction or segmentation. We made this design choice because of our belief that segmentation and feature extraction are problems without robust solutions; this belief originated from the brittleness of a preceding object recognition system [51] that relied too heavily on segmentation.

By deciding that our algorithm could not rely on accurate segmentation, we increased the possibility of false matches in our surface matching algorithms. Consequently, during development, we designed our algorithms to determine at run-time which matches are correct and which matches are incorrect. Our surface matching engine investigates many points in the scene which results in a large number of matches between models and scene. Then, correct and incorrect matches are differentiated based on the properties of the group of matches (e.g., highest similarity measure, average geometric consistency). Our approach to matching employs more brute force than matching algorithms based on feature extraction, but it results in more robust matching. This approach to matching is not limited to matching surfaces with spin-images. It can also be applied to other matching problems in computer vision including appearance-based object recognition in cluttered scenes and medical image registration.

Many object recognition systems facilitate matching by making assumptions about the shape and appearance of objects to be matched. These assumptions restrict the set of objects that can be recognized. For example, in our preceding object recognition system, we represented objects as sets of planar and quadric surface patches. We found that this object representation was too restrictive, even for recognizing man-made objects, because many objects surfaces do not

belong to the class of planar and quadric surfaces.

When developing the spin-image representation, we were searching for a representation that could model objects of almost any shape. Our starting point was to represent objects using surface meshes since they can represent the shape of most objects. Surface meshes represent shape, but they are not amenable to surface matching. Therefore, to augment surface meshes with a representation for matching, we used concepts from geometric hashing to generate the spin-image. Since a spin-image is just a transformation of a surface mesh, no restrictive assumptions about surface shape are made. By developing the spin-image representation, we learned that it is possible to match surfaces without making restrictive assumptions about surface shape.

Appendix A

Control of Mesh Resolution

Polygonal meshes[1] are common representations in computer graphics because they can represent surfaces of almost any topology and complexity. The recent growth in the availability of reliable 3-D sensors and sensing algorithms has made 3-D data much more common in computer vision research and algorithms. Because polygonal meshes are general representations for describing 3-D data, they are becoming as common in computer vision as they are in computer graphics. However, the uses of polygonal meshes by computer vision researchers may be quite different from those of computer graphics researchers. For instance, a graphics researcher may be interested in rendering a complex object represented as a polygonal mesh quickly and accurately, while a computer vision researcher may be interested in aligning two 3-D views of an object represented as polygonal meshes. In each field, the uses of polygonal meshes drives the operations that are commonly applied to them. Since the use of polygonal meshes is concentrated in computer graphics, the tools for manipulating polygonal meshes are tailored to the needs of computer graphics researchers. There is a conspicuous absence of tools for operating on polygonal meshes that are designed to aid computer vision researchers.

A particular operation that is necessary for computer graphics and computer vision is the control of resolution of polygonal meshes. The resolution of a polygonal mesh determines the amount of surface detail the mesh contains and is closely related to the number of vertices, edges and faces in the mesh. A coarse resolution mesh will contain a small number of vertices, while a fine resolution mesh will contain a large number of vertices. However, because com-

[1] "A polygonal mesh is a collection of edges, vertices and polygons connected such that each edge is shared by at most two polygons. An edge connects two vertices, and a polygon is a closed sequence of edges. An edge can be shared by two adjacent polygons and a vertex is shared by at least two edges." (Foley et al.[30])

puter graphics is primarily concerned with accurately rendering objects, and computer vision is primarily concerned with measuring properties of sensed objects, the ways in which resolution is controlled for computer graphics and computer vision will be different.

In computer graphics, the control of mesh resolution is termed mesh simplification or mesh decimation. The main motivation for mesh simplification is to reduce the number of faces describing an object, while preserving the shape of the object, so that the object can be rendered as fast as possible. Shape preservation and face reduction are the two competing forces behind mesh simplification. Some applications of mesh simplification include: removal of coplanar and adjacent faces from existing models [42] [54], creation of mesh hierarchies for level-of-detail rendering [24], and geometric compression [44] for storage and transmission of polygonal mesh models. Heckbert and Garland [39] have written a comprehensive overview of mesh simplification algorithms for computer graphics.

In contrast, an algorithm that controls the resolution of a mesh for computer vision applications should, in addition to reducing the number of faces while preserving the shape of the object, distribute the vertices of the mesh evenly over the surface of the object. The reason for this is as follows. A common operation in computer vision is to measure local properties of data, for example, calculating the gradient in a 2-D image at every pixel in the image or calculating the principal curvatures at every pixel in a range image. In these examples, an image is given, so "local" is determined by the parameterization of the image (i.e., adjacency of pixels). Unfortunately, for a polygonal mesh, locality is not well defined because a unique parameterization of the surface data does not necessarily exist. Examples of situations where well defined locality is needed in 3-D computer vision are point based registration of surface meshes [14][48], establishment of control points for computation of spline surfaces on a surface mesh [56], clustering of points for segmentation of range images [29], and calculation of surface normal [86].

A computationally efficient way to define locality is through the connectivity of vertices in the mesh. Vertices that are connected by an edge are local; however, this definition is problematic when the distance between connected vertices varies drastically, because the notion of locality cannot be associated with a fixed distance between vertices. Therefore, to ensure that local measurements on a polygonal surface mesh using connectivity are meaningful, a method for normalizing the distance between connected vertices must be developed. Given such a method,

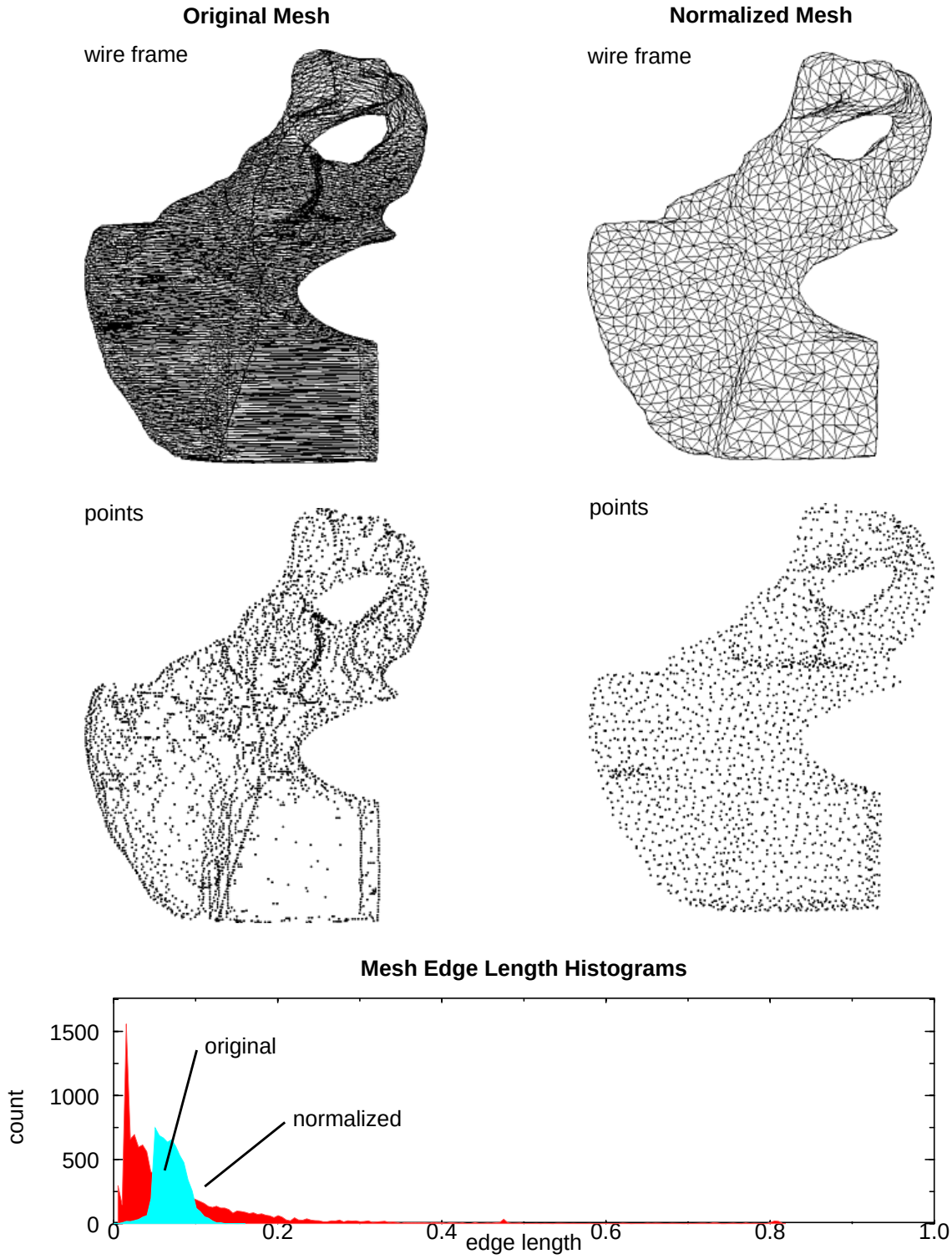


Figure A-1: Demonstration of control of resolution. An initial mesh generated from CT contours of a pelvis bone phantom with holding block creates a mesh with very long edges and very short edges. Our algorithm generates a normalized mesh with edge lengths that are centered around the desired resolution. The histogram of edge lengths shows that the normalized mesh has a much smaller spread in edge lengths than the original mesh.

by choosing the normalization distance appropriately, a mesh of any resolution can be generated. To our knowledge, no mesh simplification algorithms to date have been designed with the even distribution of vertices in mind. In this paper, we present a mesh simplification algorithm designed to normalize the distance between vertices on a surface mesh while preserving object shape.

By controlling the normalization distance, this algorithm can produce meshes of arbitrary resolution facilitating the creating of mesh hierarchies useful in coarse-to-fine approaches to 3-D computer vision.

An example of application of our algorithm to a surface mesh generated from CT contours of a pelvis bone phantom with holding block, is shown in Figure A-1. The lengths of the edges in the original mesh are widely distributed, but after application of our algorithm, the edges are compactly centered around the desired resolution. This is shown qualitatively through views before and after normalization, and quantitatively through histograms of edge lengths.

In addition to normalizing the distance between vertices, there are other requirements for any algorithm designed to control resolution of polygonal meshes used in computer vision. Below is a list of all of the necessary requirements:

- Preserve shape

  In general, the shape of the objects imaged is the property that is measured and compared in 3-D computer vision. Therefore, our algorithm must preserve shape if it is to produce meshes usable by 3-D computer vision algorithms.

- Normalize distances between vertices

  As stated above, our algorithm needs to make the distances between vertices regular so that mesh connectivity can be used to define the local neighborhood of a vertex. Normalizing distances also ensures that vertices are regularly distributed over the surface of mesh, making it possible to compute local properties evenly over the mesh. Furthermore, by normalizing the distance between vertices to a particular value, the resolution of the mesh can be controlled and hierarchies of meshes of increasing resolution can be generated.

- Minimize number of vertices

  In 3-D computer vision, the amount of computation is often proportional to the number of vertices or data points. Therefore, to minimize computation, our algorithm should minimize the number of vertices in the mesh while still meeting shape preserving and distance normalization requirements.

- Handle free-form and polyhedral shapes

  For our algorithm to be general, it must handle free-form (smooth, curved objects) and polyhedral objects because both forms are likely to be imaged.

- Handle mesh boundaries

  The meshes being simplified in computer vision often have boundaries (e.g., a polygonal mesh generated from a single range image with range discontinuities), so our algorithm must be able to control the resolution of the mesh along its boundaries as well as in its interior.

Our algorithm meets all of these criteria, given that the input surface mesh is simple (i.e., edges can be adjacent to no more than two faces) and contains only triangular faces. Most surface reconstruction algorithms create simple meshes, so the first condition is generally met in computer vision contexts. If the second condition is not met, it is trivial to transform a mesh with non-triangular faces into one composed of triangular faces by breaking each planar polygonal face into multiple separate triangular faces using a constrained planar triangulation.

After describing some related algorithms that are representative of most mesh simplification algorithms developed to date, we will describe our mesh simplification algorithm. In particular, we will detail a new mesh edge-collapse criterion that preserves object shape while normalizing the lengths of edges in the mesh along with a novel way of placing vertices after edge-collapse. We will also detail our method for propagation shape change errors; this method allows us to place a global bound on the maximum change in shape of a simplified mesh. We will follow our algorithmic discussion with a presentation of results from multiple 3-D sensing modalities including range images, marching cubes, computed tomography and digital elevation maps.

A.1 Related Work

Ideally, one would like to determine the mesh that contains the least amount of vertices and faces while conveying the shape of an object within some error bound of the original mesh. Unfortunately, the space of possible meshes is so large that searching for the globally best mesh is impractical. Therefore, most simplification algorithms to date are greedy, iterative algorithms that search for the best mesh by taking the current best step toward the global minimum. Like many mesh simplification algorithms, our algorithm is based on the iterative application of local mesh operations to transform the original mesh into a mesh meeting our simplification criteria. Some examples of mesh operations are edge-collapse, edge-split, edge flip and point removal, followed by re-triangulation. There exist many published mesh simplification algorithms; below, we describe three prominent and representative algorithms. For a more comprehensive overview, see [39].

The general flow of iterative simplification algorithms is as follows: First, order mesh primitives (vertices, edges or faces) for simplification. Next, select the best mesh primitive for simplification and apply a mesh operation to that primitive. Then, update the shape of the mesh and reorder mesh primitives for simplification, if required. Repeat application of mesh operations until no more mesh primitives can be simplified. The differences in iterative simplification algorithms come from the primitives used in simplification, how the primitives are ordered for simplification, how the mesh shape changes when a mesh operation is applied, and what criteria are used to stop mesh simplification.

One of the first iterative mesh simplification algorithms was proposed by Schroeder et al. [76]. In this algorithm, vertices are the primitives used for decimation; they are removed from the mesh, and the local neighborhood surrounding the point is re-triangulated in the local plane of the vertex. A point is removed if its distance to the best fit plane of the surrounding points is small. The primitives are not ordered; all vertices that have a planar fit error that is less than a threshold, and meet topology preserving checks, are removed. In a later version of the algorithm [78], the shape change in the mesh is limited by placing a global bound on the maximum allowable change in mesh shape. By using the point removal mesh operation, Schroeder's algorithm must shrink convex regions in the mesh and expand concave regions in the mesh. If the primitive with lowest planar fit error is removed at each iteration, the global change in

shape will be kept as small as possible. In Schroeder's algorithm, primitives are not ordered for decimation, so a vertex with a greater planar fit error can be removed before a vertex with a lesser planar fit error. Therefore, the global change in shape will not be kept as small as possible and the resulting reduction of number of points in the mesh will not be as great as it could have been had the vertices been ordered for decimation. Finally, the re-triangulation step in point removal is time consuming and complicated,

Guéziec's [35] mesh simplification algorithm improved on Schroeder's algorithm in many ways. Guéziec uses edges as the mesh primitive and edge-collapse to eliminate re-triangulation from the mesh simplification algorithm. The edges are ordered based on edge length, and a single pass through the edges is performed. During edge-collapse, a new vertex is created. Guéziec intelligently places the new vertex in a position that preserves the volume of the object in the local neighborhood of the edge, preventing drastic changes in the shape of the object. Checks on topology preservation and creation of compact triangles are also used to preserve the shape of the mesh. Guéziec's algorithm is designed primarily for relatively noise free, smoothly varying surfaces and does not have any explicit control of the resolution of the meshes generated. Furthermore, there is no explicit handling of the vertices along the boundary of a mesh to prevent shrinking while ensuring simplification.

Hoppe et al. [43] use edge-collapse, edge-swap and edge-split to iteratively refine an initial mesh that is close to a set of 3-D data points. They use a global optimization procedure that attempts to find meshes that fit the input data and have a small number of vertices and edges that are not too long. They use a spatially random ordering of edges during simplification and perform a fixed number of iterations. Since they fit new vertex positions based on supplied data, the shape of the object is preserved. Their algorithm can handle free-form as well as polyhedral objects. Their optimization procedure requires three nested loops and is consequently quite slow, but it does produce very concise and accurate meshes. The inclusion of a term that penalizes long edges in the optimization is a step toward controlling the overall distribution of vertices in the surface mesh.

There exist some alternative, particle-based, approaches to generating a regular sampling of points on a surface. Turk [92] presents an algorithm that simplifies an initial mesh as follows. A small number of points are placed on the surface mesh and then pushed around the faces of

the surface mesh by repulsive forces until they are evenly distributed. Once distributed, the points are added to the initial mesh and then all of the vertices in the initial mesh are iteratively removed. The end result is an even sampling of the original surface mesh. This algorithm will work best for smoothly varying surfaces and is quite complicated. Witkin and Heckbert [98] present an algorithm for obtaining a regular sampling of an implicit surface using a particle-based approach. They are able to control the sampling of an implicit surface by changing the properties of the forces controlling the particles and can maintain a regular sampling even while the surface is rapidly changing. However, their algorithm relies on an underlying implicit surface and does not address the issue of mesh generation. Shimada [79] presents an algorithm for controlling the sampling of 2-D and 3-D meshes using a physically-based triangulation method called the bubble mesh. His algorithm creates uniformly spaced vertices but, when triangulating 3-D surfaces, requires that the surface have a 2-D parameterization. In the case of general surfaces, this requirement cannot be guaranteed.

A.2 Algorithm for Control of Mesh Resolution

For an arbitrarily shaped object, it is impossible to make the distance between vertices exactly the same for all vertices in the mesh while still adequately describing the shape of the object. Therefore, we quantify the spacing between vertices using local and global statistics on the histogram of lengths of edges in the mesh. An example of an edge length histogram is given in Figure A-2. We define mesh *resolution* to be the median of the lengths of all of the edges in the mesh, and we define the *edge length spread* to be the upper quartile of edge lengths minus the lower quartile of edge lengths (i.e., the half width) in the edge length histogram. Given these definitions, the goal of our algorithm is to adjust the resolution of the original mesh to a desired resolution while minimizing the edge length spread of the histogram. We call this process *length normalization*.

An additional constraint on length normalization is that the original shape of the object must be preserved; we assume that the original shape of the object is given by the mesh input into the algorithm. In our algorithm, two operations are iteratively applied to edges in the mesh to obtain the mesh of the desired resolution: edge-split is used to remove long edges, while edge-collapse is used to remove short edges. During edge-split, an edge is broken at its midpoint into

two edges; the edge-split operation does not change the shape of the mesh. During edge-collapse, an edge is reduced to a point, so the local shape of the mesh is modified. In our algorithm, the position of the point resulting from edge-collapse is chosen to preserve the shape of the object, but some change in shape is unavoidable when edges are removed from the mesh. However, the change in shape of the mesh can be minimized at each iteration by intelligently ordering the edges for operation. More specifically, the edges in the mesh are ordered for operation by measuring the change in shape that results from application of each edge operation (*shape change measure*). Using this ordering, the edges that change the shape of the mesh the least are applied first, thus minimizing the change in shape of the mesh at each iteration. This best-first approach produces meshes that preserve shape better during length normalization than would an algorithm that chooses edges randomly.

To strike a balance between preserving shape and normalizing the lengths of the edges in the mesh, the order of application of edge operations is also made a function of the length of the edge being operated on. More specifically, an *edge length weight* is computed for each edge; the order in which an edge is operated on is then determined by the product of its edge length weight and its shape change measure. For example, with this ordering, given two edges with the same shape change measure whose lengths are shorter than the desired resolution, the shorter edge will be collapsed first.

We prevent the shape of the mesh from changing too much by placing a bound on the maximum allowable change in mesh shape. Each time an edge operation is applied, the edge's shape

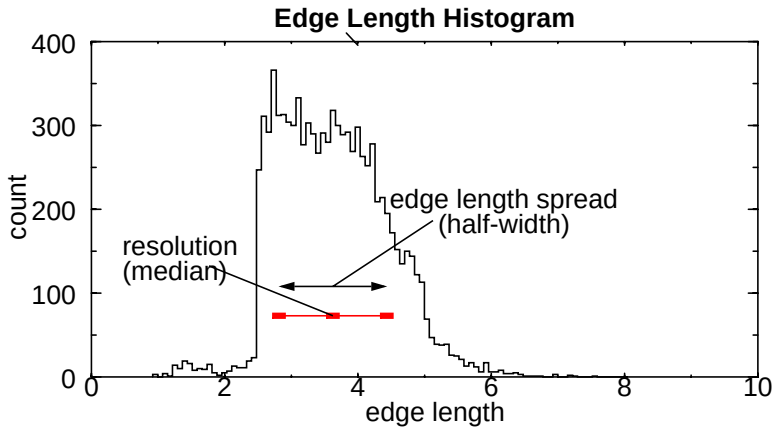


Figure A-2: Histogram of edge lengths. The resolution of a mesh is the median of its edge length histogram and the edge length spread is the half-width (upper quartile minus lower quartile) of the histogram.

change measure is added to the *accumulated shape change* (the accumulation of the change in shape that the edge has undergone in previous iterations) of all of the edges in the neighborhood of the edge. If the accumulated shape change of an edge is made greater than the maximum allowable change in shape, the edge is removed from consideration. This ensures that, over the entire surface of the mesh, the change in shape remains below a user-defined bound. Furthermore, if the length of an edge is within some user defined bounds of the desired mesh resolution, the edge will not be decimated. The limit on the maximum allowable shape change and the length of edges being within some bounds of the desired resolution will eventually prevent all edges in the mesh from being decimated. This constitutes the stopping criterion for the algorithm. Since our algorithm attempts to normalize lengths while preserving shape, most, but not all, of the edge lengths will be within the desired bounds.

A.2.1 Definitions

Before describing the mesh normalization algorithm in detail, some definitions need to be established. Edge-collapse and edge-split operate on local neighborhoods in the mesh. The exact definitions of the local neighborhoods of an edge and vertex are as follows. Let the *EdgeStar(e)* be the local neighborhood in the mesh affected by the edge-collapse operation on an edge *e*. *EdgeStar(e)* contains all of the faces that contain at east one of the vertices making edge *e*, as well as the edges and vertices that make these faces. Let the *VertexStar(v)* be the local mesh neighborhood of the vertex *v* created when an edge-collapse operation is applied to an edge. It contains all of the faces that contain the vertex *v* along with the edges and vertices making up these faces. Illustrations of *EdgeStar(e)* and *VertexStar(v)* are given in Figure A-4. Let the *EdgeDiamond(e)* be the local neighborhood in the mesh affected by the edge-split operation on an edge *e*. It contains the faces that are adjacent to the edge *e*. The edges and vertices that make these faces are also in *EdgeDiamond(e)*. Let the *VertexDiamond(v)* be the local mesh neighborhood of the vertex *v* created when an edge-split operation is applied to an edge. It contains the four faces that contain the vertex *v* along with the edges and vertices making up these faces. Illustrations of *EdgeDiamond(e)* and *VertexDiamond(v)* are given in Figure A-7.

A.2.2 Overview of Algorithm

Input into length normalization is a desired resolution L_0 and the acceptable deviation in length

L_D from the desired resolution for edges in the normalized mesh. The upper and lower bounds on edge lengths are then

$$L_{MIN} = L_0 - \frac{L_D}{2} \qquad L_{MAX} = L_0 + \frac{L_D}{2} \tag{A.1}$$

In addition, the maximum allowable change in shape C_{MAX} is input to the algorithm.

Given in detail, our algorithm is as follows: First, a priority queue (a dynamically ordered queue) is created from all the edges in the mesh. The position of an edge in the priority queue is the product of a edge length weight and the shape change measure of the edge. Next, the first edge in the priority queue is popped off the queue and operated on. If the length of the edge is greater than L_{MAX}, the edge is split at its midpoint. This split changes the neighborhood of the edge by adding an edge, a vertex and two new faces. If the edge length is less than L_{MIN}, the edge is collapsed into a point, changing the neighborhood of the edge by eliminating a vertex, two faces and an edge. When an edge is collapsed, its shape change measure is added to the accumulated shape change of the edges in the new neighborhood of the edge. After the operation is applied, the edges in the old neighborhood of the edge are removed from the priority queue Then, the edges in the new neighborhood of the mesh are added to the priority queue if they meet the following criteria: their lengths are outside the edge length bounds L_{MAX} and L_{MIN}; their accumulated shape change is not greater than C_{MAX}; and they meet additional checks that prevent changes in topology and shrinkage of the mesh boundary. Edges are iteratively popped off the queue and operated on until no more edges exist in the queue. A pseudo-code description of the length normalization algorithm is given in Figure A-3.

A.2.3 Shape Change Measure

The shape change measure of an edge is defined as the distance between the current mesh and the mesh that results from operating on the edge. During length normalization, we want to place a bound on the maximum change in shape of the mesh. Therefore, our shape change measure is defined as the maximum distance between meshes before and after an edge operation is applied. Since edge operations affect only a local neighborhood of the edge, the distance between meshes can be measured by comparing only the local mesh neighborhoods before and after application of the operation.

We consider mesh shape to be conveyed by the faces of the mesh (not just the vertices), so an accurate measure of distance between meshes must consider distance between mesh faces. We define the asymmetric distance between a mesh M_1 and a mesh M_2 to be the maximum of the distance between any point on M_1 and its associated closest point on M_2. Because meshes are composed of subsets of linear elements (points, lines and planes), the maximum distance between M_1 and M_2 will occur between a vertex of M_1 and a face of M_2. Therefore, the distance between M_1 and M_2 can be defined as the maximum Euclidean distance between a vertex v_i of M_1 and its closest point, $v_{closest}$, on the closest face f_j of M_2 to v_i.

$$d(M_1, M_2) = \max_{v_i \in M_1} \left(\min_{f_j \in M_2} \left\| v_i - v_{closest}(v_i, f_j) \right\| \right) \tag{A.2}$$

```
NormalizeLengths(LMIN,LMAX,CMAX,MESH)
    // Initialize edge priority queue
    PQ = InitializePriorityQueue()
    For all edges e in MESH
        C = ShapeChangeMeasure(e,MESH)
        W = EdgeLengthWeight(LMIN,LMAX,e,MESH)
        AccumulateShapeChange(e) = 0
        If (CanOperateOn(e,LMIN,LMAX,CMAX,MESH))
            InsertEdgeInPriorityQueue(e,W*C,PQ)
    // Normalize mesh
    While (!Empty(PQ))
        edge e = PopPriorityQueue(PQ)
        If (Length(e)>LMAX)
            For all edges oe in EdgeDiamond(e)
                RemoveEdgeFromPriorityQueue(oe,PQ)
            vertex v = SplitEdge(e,MESH)
            For all edges ne in VertexDiamond(v)
                AccumulateShapeChange(ne) = AccumulateShapeChange(ne) + 0
                C = ShapeChangeMeasure(ne,MESH)+AccumulateShapeChange(ne)
                W = EdgeLengthWeight(LMIN,LMAX,ne,MESH)
                If (CanOperateOn(ne,LMIN,LMAX,CMAX,MESH))
                    InsertEdgeInPriorityQueue(ne,W*C,PQ)
        If (Length(e)<LMIN)
            For all edges oe in EdgeStar(e)
                RemoveEdgeFromPriorityQueue(oe,PQ)
            vertex v = CollapseEdge(e,MESH)
            For all edges ne in VertexStar(v)
                AccumulateShapeChange(ne) = AccumulateShapeChange(ne) +ShapeChange(e)
                C = ShapeChangeMeasure(ne,MESH)+AccumulateShapeChange(ne)
                W = EdgeLengthWeight(LMIN,LMAX,ne,MESH)
                If (CanOperateOn(ne,LMIN,LMAX,CMAX,MESH))
                    InsertEdgeInPriorityQueue(ne,W*C,PQ)
```

Figure A-3: Pseudo-code description of length normalization algorithm.

The closest point on a triangle to a point in space is computed by first projecting the point along the triangle normal onto the plane defined by the triangle. If the projected point lies inside the triangle, then it is the closest point. Otherwise the point is projected perpendicularly onto the lines that determine the edges of the triangle. If the point projects onto the lines between two vertices of the triangle, then this is the closest point. Otherwise, the closest point is one of the vertices of the triangle.

The distance $d(M_1,M_2)$ is not symmetric, so we define a distance metric between two meshes $D(M_1,M_2)$ to be the maximum of $d(M_1,M_2)$ and $d(M_2,M_1)$.

$$D(M_1, M_2) = max(d(M_1, M_2), d(M_2, M_1)) \quad \text{(A.3)}$$

In our length normalization algorithm, we use $D(M_1,M_2)$ as our shape change measure. Intuitively, the shape change measure will be zero when the surfaces described by the faces of the mesh coincide, even when the faces, edges and vertices of the two meshes are not exactly the same. Using the maximum distance between meshes as our shape change measure gives our algorithm the ability to operate on edges along surface shape discontinuities, like ridges and corners as long as the distance between meshes remains small during operation. Operating on ridges and corners is not possible with most mesh simplification algorithms because the shape change measures used (e.g., distance to local tangent plane in [76]) are over cautious and prevent simplification along surface shape discontinuities even when simplification will not change the shape of the mesh.

In the general case of comparing two meshes, computing the $D(M_1,M_2)$ is computationally expensive. However, as will be shown in the next section, computing the shape change measure between meshes before and after application of an edge operation is computationally feasible since there the change in mesh shape is restricted to a local neighborhood of the mesh.

A.2.4 Edge Operations

The first edge operation we will consider is edge-collapse. As shown in Figure A-4, the effect of the edge-collapse operation is to shrink an edge in the mesh to a point, thereby removing the edge and its two adjacent faces from the mesh. Edge-collapse can be performed quickly through local mesh operations that remove the affected faces, edges and vertices from our sur-

face mesh data structure and then update the pointers between adjacent faces, vertices and edges.

An important variable in the edge-collapse operation is the position of the new vertex that results from edge-collapse; the position of the other vertices in *EdgeStar(e)* remain fixed during edge-collapse. A simple method for positioning the vertex would be to place the vertex at the midpoint of the collapsed edge. However, as shown in 2-D in Figure A-5, this simple placement of the new vertex keeps the vertex on the surface mesh, but can cause excessive shape change (shrinkage or expansion) in areas of high curvature in the mesh. Instead, we allow the new vertex to be placed off of the edge, in order to reduce the shape change measure of the edge-collapse. In particular, the position of the new vertex $\boldsymbol{v}$ is the average of the projection of the midpoint of the edge $\boldsymbol{v}_m$ onto the N planes defined by the faces in *EdgeStar(e)*.

$$\boldsymbol{v} = \boldsymbol{v}_m - \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{n}_i\boldsymbol{v}_m + d_i)\boldsymbol{n}_i \qquad \boldsymbol{v}_m = \frac{\boldsymbol{v}_1 + \boldsymbol{v}_2}{2} \tag{A.4}$$

The planes in *EdgeStar(e)* are defined by their surface normal $\boldsymbol{n}_i$ and offset d_i. As shown in Figure A-5 for a 2-D example, placing the new vertex based on projections onto the planes of the surrounding faces prevents shrinkage of the mesh by distributing the change in shape above and below the collapsed edge. This is in contrast to placing the new vertex at the midpoint of

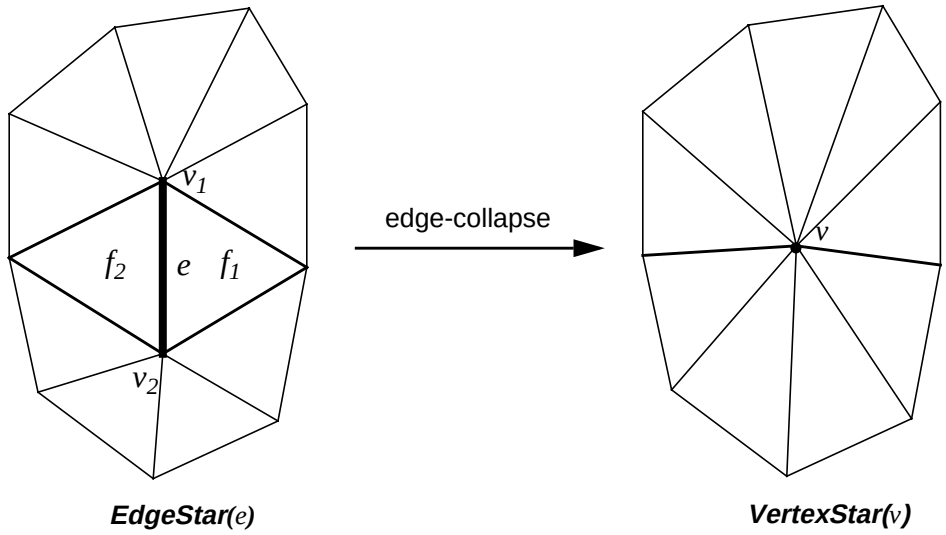


Figure A-4: The effect of edge-collapse on the local neighborhood of a mesh. The edge e is collapsed to a vertex v, eliminating edge e, faces f_1 and f_2, and vertices v_1 and v_2. The local neighborhood of e is termed *EdgeStar(e)* and the local neighborhood of the vertex v is termed *VertexStar(v)*.

the collapsed edge where the change in mesh shape is not balanced because only shrinkage occurs. Figure A-6 shows the cumulative effect of our shape preserving placement of the vertex during edge-collapse versus the placement of the vertex at the midpoint of the collapsed edge. A surface mesh model of a femur bone generated from CT contours is shown in Figure A-6. Two 2-D slices through the model are shown for the original mesh: a mesh normalized using shape preserving vertex placement and a mesh normalized using midpoint vertex placement. From the slices it is apparent that, with respect to midpoint placement, shape preserving placement reduces the shrinkage that can occur in areas of high curvature during length normalization.

The shape change measure for an edge that is going to be collapsed can be computed using just the mesh primitives in *EdgeStar(e)* and *VertexStar(v)*. After edge-collapse the vertices along the border of *VertexStar(v)* are the same as the vertices on the border of *EdgeStar(e)*. Therefore, the shape change measure can be calculated as the maximum of the distance between *v* and its closest point on the faces of *EdgeStar(e)*, v_1 and its closest point on the faces of *VertexStar(v)*, or v_2 and its closest point on the faces of *VertexStar(v)*.

$$d(VertexStar(v), EdgeStar(e)) = max\left\{\begin{array}{l} \min\limits_{f_j \in EdgeStar(e)} \left\| v - v_{closest}(v, f_j) \right\| \\ \min\limits_{f_j \in VertexStar(v)} \left\| v_1 - v_{closest}(v_1, f_j) \right\| \\ \min\limits_{f_j \in VertexStar(v)} \left\| v_2 - v_{closest}(v_2, f_j) \right\| \end{array}\right\} \quad (A.5)$$

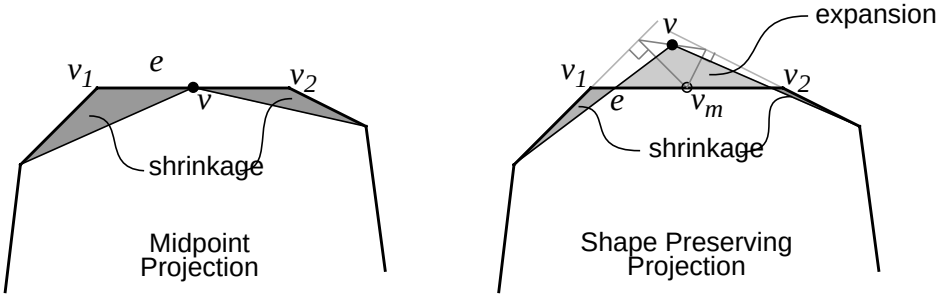


Figure A-5: Placement of the new vertex generated during edge-collapse at the midpoint of the collapsing edge causes shrinkage of the mesh during normalizations (right). However, by placing the new vertex off of the edge during edge-collapse, shrinkage and expansion are combined to limit the shape change in the mesh (left).

We use the edge-split operation to remove edges that are too long from the mesh. As shown in Figure A-7, the edge-split operation splits an edge at a vertex on that edge to produce three new edges, two new faces and a new vertex. The position of the new vertex is chosen as the mid-point of the edge being split. Since the mesh surface before and after edge-split is the same, the shape change measure for the edge-split operation is zero. Edge-split can be performed quickly through local mesh operations that add the new faces, edges and vertex to a surface mesh data structure and then update the pointers between adjacent faces, vertices and edges.

A.2.5 Accumulated Shape Change

Each time an edge is collapsed, the shape of the mesh changes slightly. The shape change measure we use is the maximum distance between the mesh before and after the edge was col-

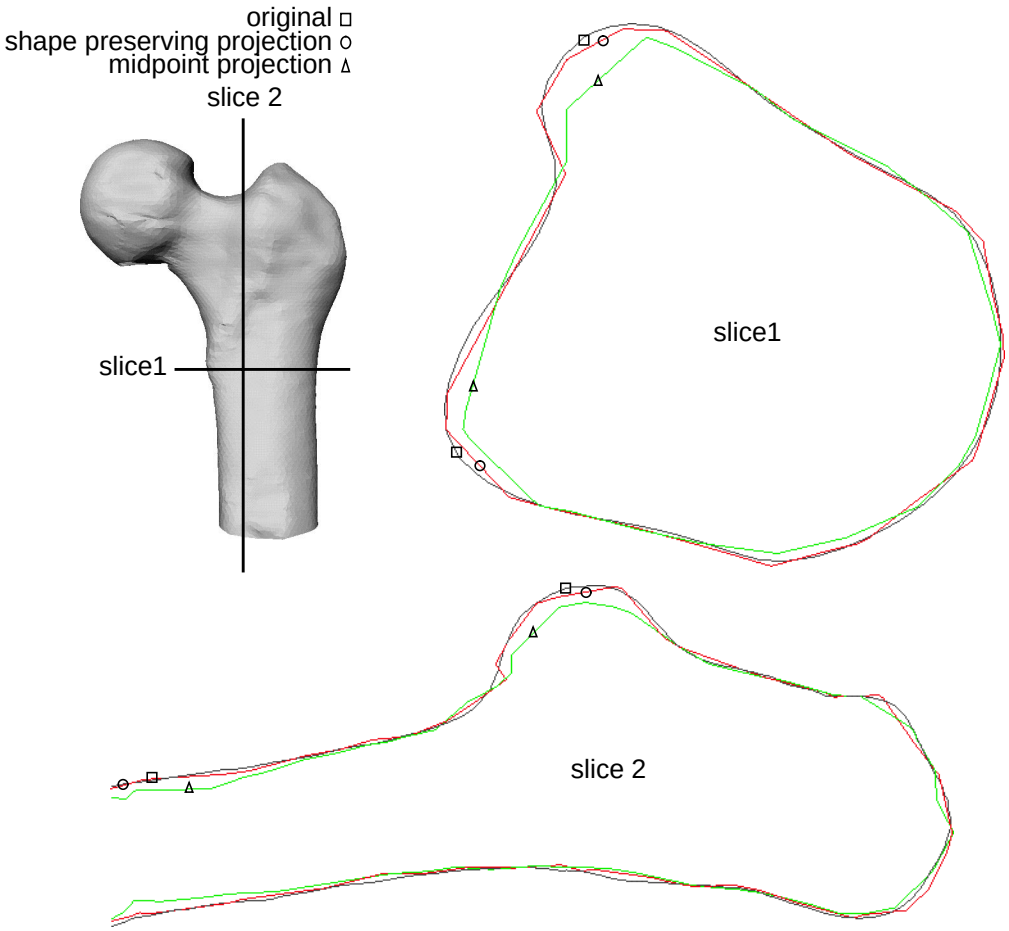


Figure A-6: Illustration of the benefit of shape preservation vertex positioning during edge-collapse. Two 2-D slices through a femur model normalized using shape preserving and midpoint positioning show that midpoint positioning of vertices during edge-collapse shrinks the surface mesh in areas of high curvature while shape preserving positioning reduces shrinkage in high curvature areas. Shape preserving projection balances shrinkage and expansion to prevent excessive shape change.

lapsed. We limit the amount of shape change that occurs during normalization by storing an *accumulated shape change* in mesh shape accrued so far by each edge during normalization. Initially each edge starts with zero accumulated shape change. When an edge *e* is collapsed, its shape change measure is added to the accumulated shape change of all of the edges in *Vertex-Star(v)*. By keeping track of the worst case change in mesh shape for each edge, we can limit the global maximum change in mesh shape. In other words, edges that have an accumulated shape change greater than a specified bound can be prevented from being collapsed. The idea of accumulating shape change is attributable to Schroeder [78]. However, his measure of shape change (distance to local best fit plane) is less accurate, and therefore more conservative, than ours. The edge-split operation does not change the shape of the mesh, and it does not increase the accumulated shape change of the edges in the neighborhood of the edge.

The global bounds on accumulated shape change are illustrated in Figure A-8. Placing a maximum bound on the total change in shape of the edges in the mesh can be visualized as two surfaces that contain the original mesh: an inner surface that bounds shrinkage and an outer surface that bounds expansion. During normalization, the global bound on accumulated shape change prevents the normalized surface from moving outside of these bounding surfaces. In Figure A-8, the bounding surfaces are expansions and contractions of the original surface mesh generated by projecting each vertex *v* in the original surface mesh along the surface normal of the best fit plane to the vertices in *VertexStar(v)*. The vertices are projected (out for outer bound and in for inner bound) a distance equal to the maximum allowable accumulated shape change C_{MAX}. Two 2-D slices through the normalized mesh and inner and outer bounding surface clearly show that the normalized surface stays within its bounds. By accumulating and limiting

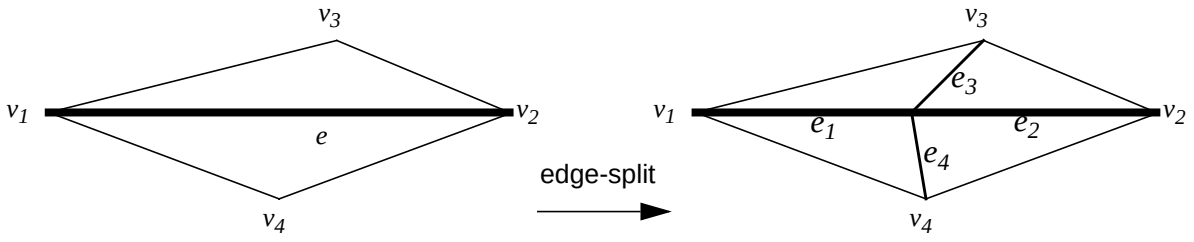


Figure A-7: The effect of edge-split on the local neighborhood of a mesh. The edge e is split at a vertex v, adding three new edges and two new faces to the mesh. The local neighborhood of e during edge-split is termed *EdgeDiamond(e)* and the local neighborhood of the vertex v after edge-split is termed *VertexDiamond(v)*.

the maximum allowable shape change, we have developed a method for controlling the total change in mesh shape.

A.2.6 Edge Ordering

During normalization, we would like to operate on edges with lengths that are far from the desired resolution in order to reduce the edge length spread. We would also like to prevent operations on edges that have large accumulated shape change in order to prevent drastic changes in mesh shape. To implement these requirements, the edges in the mesh are ordered for operation by storing them in a priority queue. The order of an edge in the priority queue is determined by the product of the accumulated shape change *C* for the edge and an *edge length weight W* for the edge; edges with a small product *C*W* will be toward the top of the queue.

The edge length weight of an edge is generated from a Gaussian of edge length

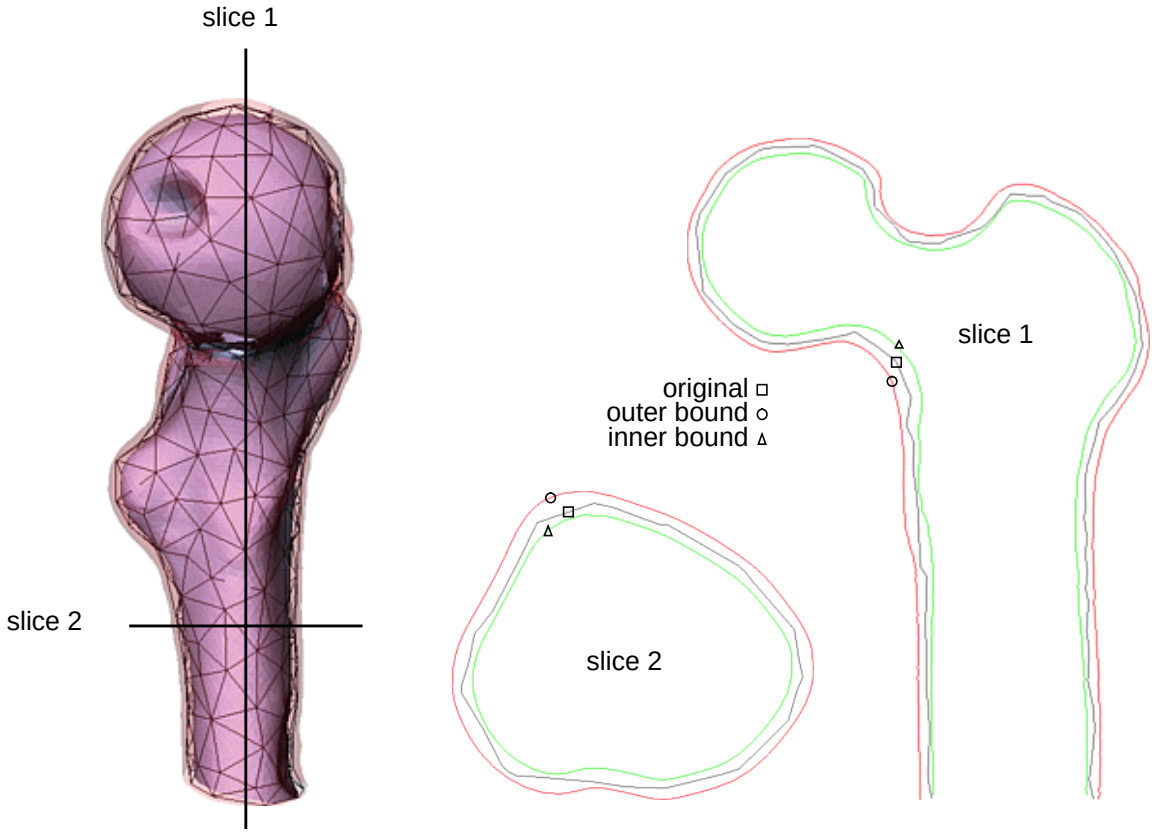


Figure A-8: Visualization of global bounds on accumulated shape change for a model of a femur. Normalization prevents excessive shape change by keeping the simplified surface mesh (wire frame, left) inside the inner and outer global error bound surfaces (shaded, transparent, left). Two 2-D slices through the normalized mesh and the inner and outer bound surfaces clearly show that the normalized mesh is within the error bound surfaces. NOTE: The bounding surfaces are for visualization only and are not used in the normalization algorithm.

$$W = \exp\left(-\frac{(l - L_0)}{L_D^{\,2}}\right) \tag{A.6}$$

where the length of the edge is l, the desired resolution is L_0, and the acceptable edge length spread is L_D. Using this edge length function will assign a small weight to edges that are much shorter or longer than the desired resolution. The accumulated shape change of an edge will be large for edges that have changed the shape of the mesh a great deal. By using the product of accumulated shape change and edge length weight, edges that are very short or very long and have not deviated from their original positions will be decimated before edges that are close to the desired resolution or that have deviated a great deal from their original position.

Using the edge length weight in addition to the accumulated shape change for ordering edges for operation is our main mechanism for generating length normalized meshes. If an edge is too long its edge length weight will be small and it will be split. If an edge is too short, its edge length weight will also be small and the edge will be collapsed. Therefore, over many iterations, the lengths of the edges in the mesh will be forced toward the desired resolution. By using accumulated shape change and not just the immediate shape change measure of an edge, edges that have been operated on a great deal (and have changed the shape of the mesh over many iterations) will be avoided. This has the effect of distributing the change in mesh shape over the entire surface of the mesh instead of concentrating the change in shape at specific places.

During normalization, edges are constantly removed and added to the priority queue. An edge can no longer be operated on, and hence will not be added to the priority queue, if its accumulated shape change exceeds the maximum allowable accumulated shape change C_{MAX} if operated on. Furthermore, an edge will not be added to the priority queue if its length is within the desired edge length bounds (L_{MIN},L_{MAX}). These two conditions eventually cause the priority queue to become empty, so the iterations on the mesh edges must stop. Since some edges will achieve the accumulated shape change bound before their length is within the edge length bounds, not all the edges in the final mesh will have lengths inside of the bounds.

A.2.7 Mesh Boundaries

In computer vision, the meshes generated from 3-D data will often have boundaries due to par-

tial views of a scene or incomplete scene data. Like interior edges, boundary edges are inserted into the priority queue based on their accumulated shape change measure and edge length weight. If the boundary edge is shorter than the desired mesh resolution, then it should be collapsed. Collapsing a boundary edge changes the shape of the mesh and the shape of the boundary. Since the boundary usually contains important information about the shape of the object (e.g., occluding contour), its shape needs to be preserved. Since our shape change measure determines the maximum distance between the mesh faces before and after collapse, it takes into account the change in shape of the mesh shape and its boundary. If collapsing an edge changes the boundary greatly, then the shape change measure of the edge is large and the edge is not operated on. As with interior edges, splitting a boundary edge does not change mesh shape, so its shape change measure is zero. The edge-collapse and edge-split operations as applied to boundary edges are given in Figure A-9.

A.3 Discussion

There exists a distinct advantage in using the maximum distance between meshes before and after operation as our shape change measure; edges along creases or ridges in the surface mesh can have small shape change measure and hence be operated on. This is in contrast to more conservative measures of shape change, such as distance to best fit plane [76]. Along ridges, the distance to best fit plane (plane fit to the vertices in the edge star of the edge) is large, so these edges cannot be decimated. Figure A-10 demonstrates the effectiveness of our shape

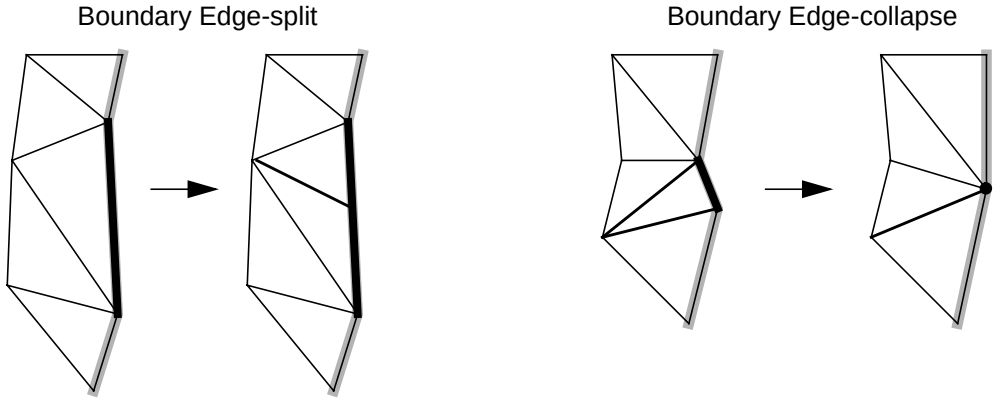


Figure A-9: The effect of the edge-split and edge-collapse operations on edges on the boundary of the surface mesh. Special version of the operations are implemented because the local mesh neighborhood of an edge on the boundary is different from the local mesh neighborhood of an edge in the interior of the mesh.

change measure for such cases. A surface mesh representation of a cube is decimated using three different mesh simplification algorithms. After the original surface mesh, the result of an algorithm (similar to Schroeder et al. [78])is shown This algorithm simplifies meshes by removing points that are a small distance to the best fit plane of the point, followed by re-triangulation of the local neighborhood. Since points on the 12 creases of the cube are a large distance to the plane fit to the local neighborhood of the point, they are not removed. The result is that a large number of vertices are left along the creases of the cube, while a few remain in the interior of the cube sides. In addition, the lengths of the edges in the surface mesh are widely distributed.

The next simplification result was generated using our algorithm without using the edge-split operation. The measure of shape change used was the average distance of the vertices in edge *e* from the best fit plane to *EdgeStar(e)*. The edges were ordered based on this distance to the best fit plane and edge length. Although there is less spread in edge length, the edges along the cube creases are not collapsed because the distance to the best fit plane is large along the creases. The edges along the creases show up in the edge length histogram as the spike at the short edge end of the histogram.

The final result shows normalization of the cube lengths using the algorithm presented in this paper. Using the distance between meshes before and after operation as the shape change measure allows edges along the creases of the cube to be operated on. The result is a much smaller spread in edge lengths than would be possible with the previous two implementations. This result clearly shows the ability of our algorithm to normalize surface mesh representations of polyhedral objects.

Figure A-11 demonstrates, in the extreme, how much the distance between meshes shape change measure increases the simplification over that possible when using the planar fit error shape change measure. Using the distance between meshes criterion, the cube mesh shown at the top of Figure A-10 can be reduced to 26 vertices, while the planar fit error criterion only allows a reduction to 309 vertices. In both cases, the algorithms are executed until no more simplification is possible without distorting the shape of the cube.

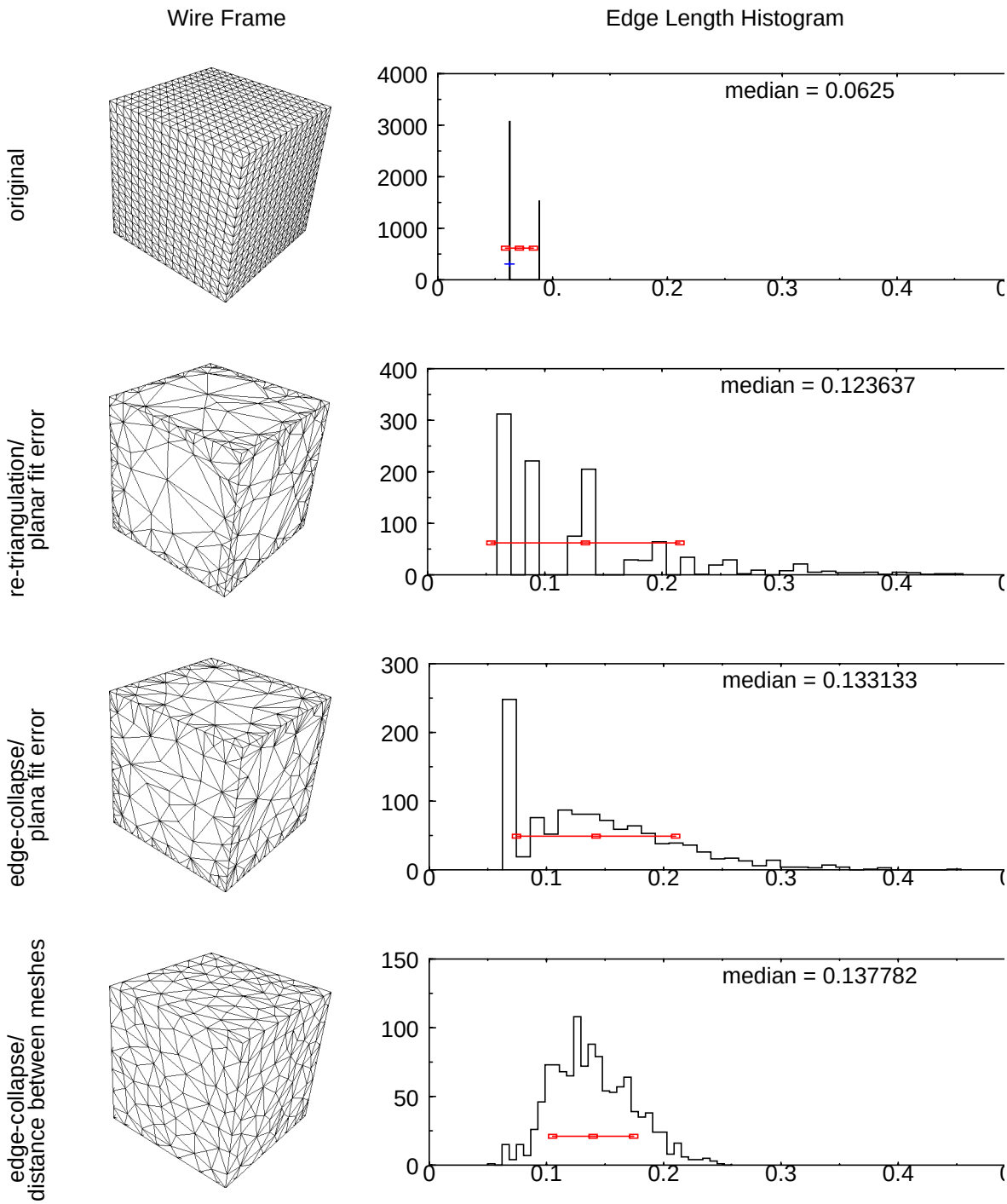


Figure A-10: Comparison of length normalization for three mesh simplification algorithms. Mesh decimation using point removal followed by re-triangulation and a planar fit error results in a large spread in edge lengths and too many vertices on the creases of the cube. An algorithm that uses edge-collapse and planar fit error has a smaller spread in edge lengths, but still does not remove edges on the creases of the cube. Not until the distance between meshes is used to measure shape change are edges removed along the creases of the cube, resulting in a much more compact edge length histogram.

Overall, the computation complexity of length normalization algorithm is *O(NlogN)* where *N* is the number of edges in the original mesh. Creating the priority queue of edges takes *N* insertions each into a dynamically sorted list (the priority queue). The priority queue is implemented efficiently as a Fibonacci heap [66], so each insertion takes *O(logN)* time. During normalization, each edge operation requires the re-insertion of a roughly fixed number of edges back into the priority-queue. If *M* edge operations are applied to the mesh, mesh normalization will take *O(MlogN)*. In our experience, the number of edge operations is on the order of the number of edges in the original mesh, so application of all of the edge operations takes *O(NlogN)*. Combining this with the time it takes to create the priority queue, the overall complexity of the algorithm is *O(NlogN)*.

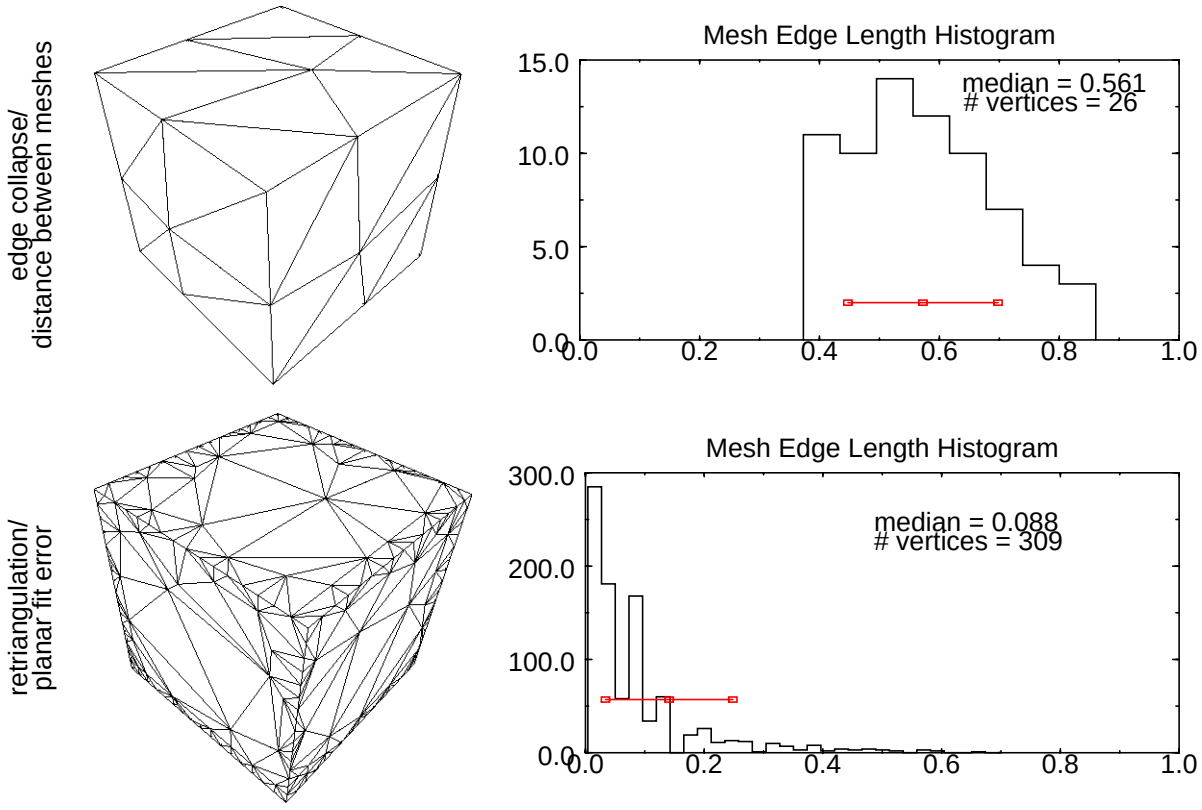


Figure A-11: Comparison of shape change measure on the amount of simplification possible for polyhedral objects. Both of the cubes shown above were simplified as much as possible without deviating from the original cube shape (shown at the top of Figure A-10), using two different shape change measures. Using the distance between meshes shape change measure allows for a much greater simplification of the cube than is possible with the planar fit error shape change measure because planar fit error prevents simplification along the creases edges in the cube.

A.4 Results

In order to demonstrate the generality of our algorithm, we present results from multiple sensing domains common in 3-D computer vision. The results are represented as hierarchies of surface meshes generated from the original data set. Each level in a hierarchy is generated by applying the length normalization algorithm to the original data. The resolution of each level is set by adjusting the edge length bounds (L_{MIN},L_{MAX}) and the maximum accumulated shape change C_{MAX} input into the algorithm. The edge length bounds input into the algorithm are shown as a line with boxes indicating the bounds and the desired resolution on the edge length histogram that accompanies each level of the hierarchy. The desired resolution doubles between each level of the mesh. This is validated by the doubling of the measured median of edge lengths between each level. The edges not within the edge length bounds have accumulated shape change that is greater than C_{MAX}. In the results, the ability of our algorithm to normalize edge lengths is shown visibly with wire frame meshes with hidden lines removed and shaded surface meshes.

Figure A-12 shows a hierarchy of normalized surface meshes for a model of a rubber ducky. The input surface mesh was generated using a volumetric range image merging algorithm [97]. Multiple views of the ducky taken with a structured light range sensor were inserted into a volumetric data structure that describes the surface of the duck. The seamless surface of the duck was then extracted from the volume using the Marching Cubes algorithm. A characteristic of Marching Cubes is the generation of many short edges; these short edges generate the aliasing noticeable in the original data. The first level of the hierarchy removes these short edges and subsequently the aliasing. This result demonstrates the ability of our algorithm to handle Marching Cubes data and curved surfaces without holes.

Figure A-13 shows a hierarchy of normalized surface meshes for a model of a femur bone. The original data was created from Computed Tomography (CT) slices of a human femur bone. Surface contours were extracted from each CT slice, and the surface contours were subsequently linked to create the surface mesh. There are holes at the top and bottom of the femur bone due to incomplete 3-D data. This result demonstrates the ability of our algorithm to produce normalized surface meshes while preserving shape from surface meshes constructed from CT contours that contain boundaries.

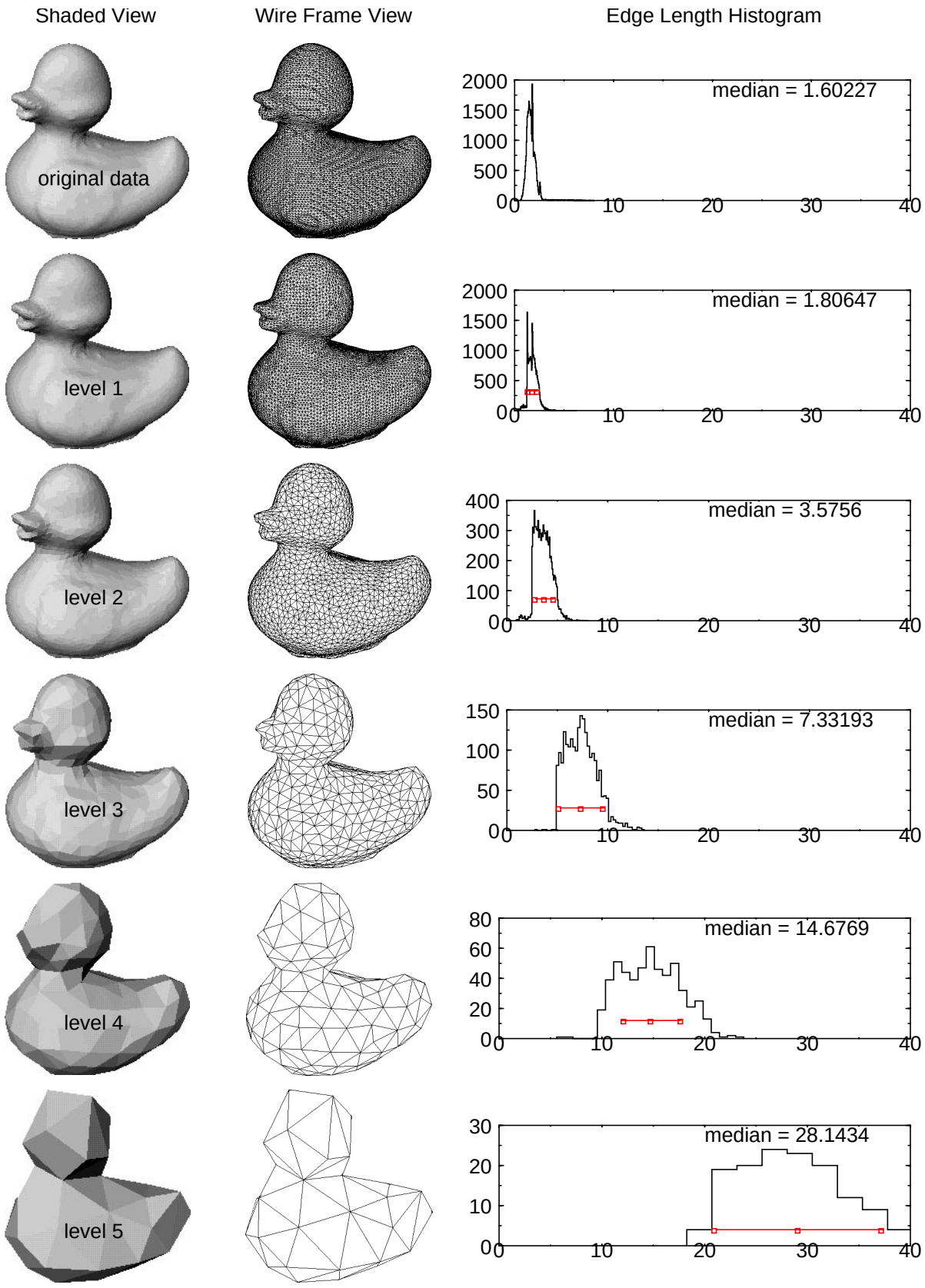


Figure A-12: Hierarchy of duck meshes.

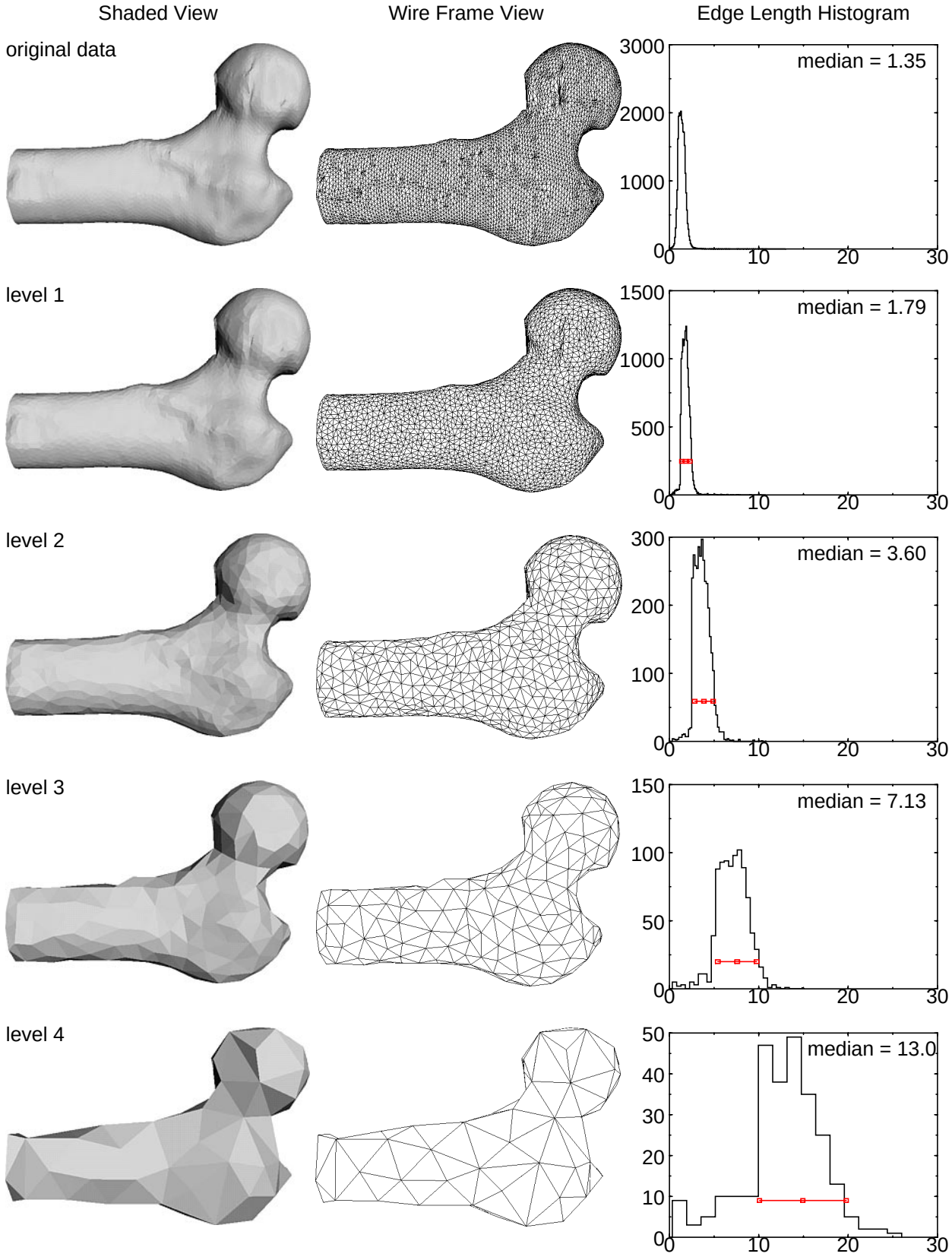


Figure A-13: Hierarchy of femur meshes with original data from CT.

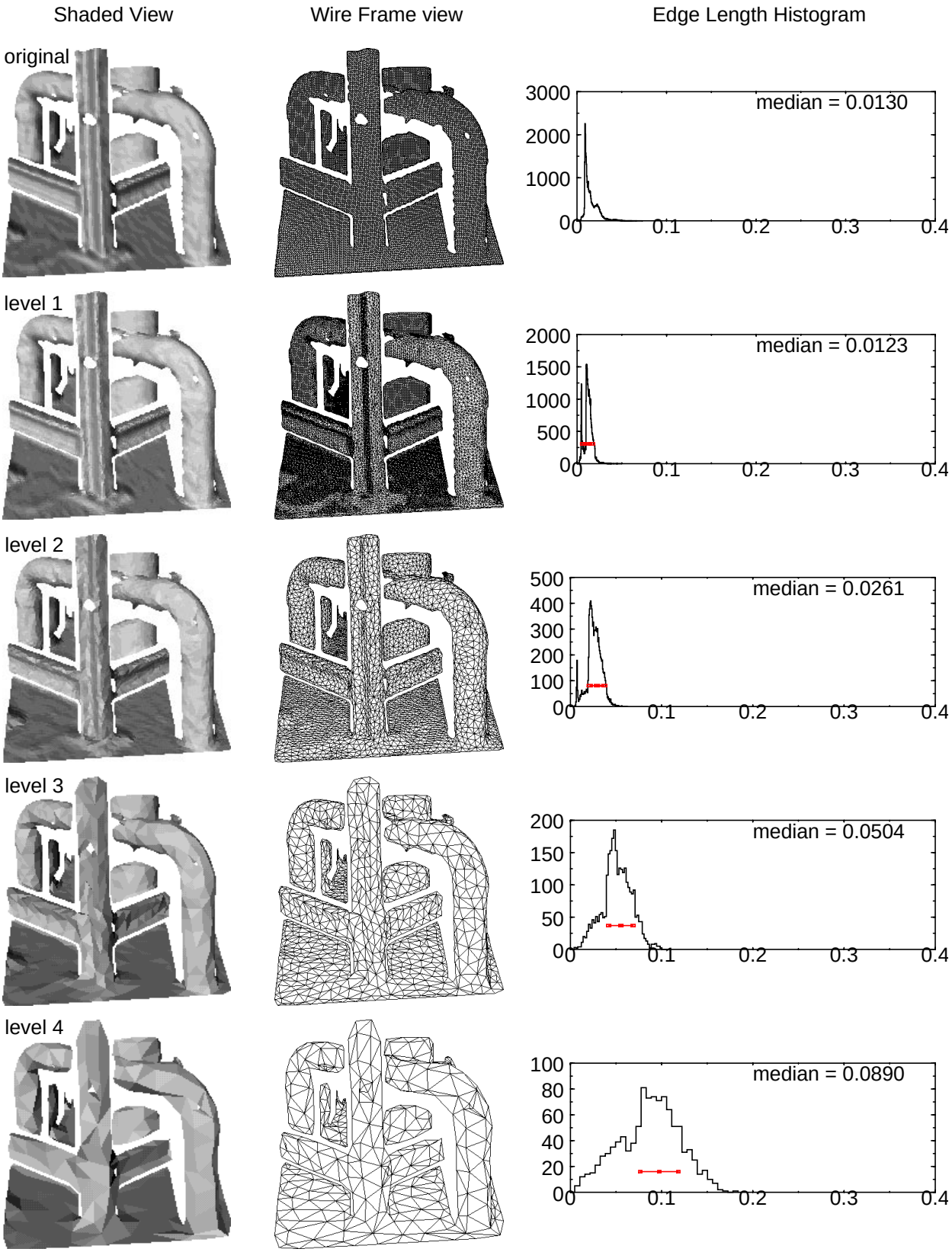


Figure A-14: Hierarchy of range image with edges along range discontinuities removed.

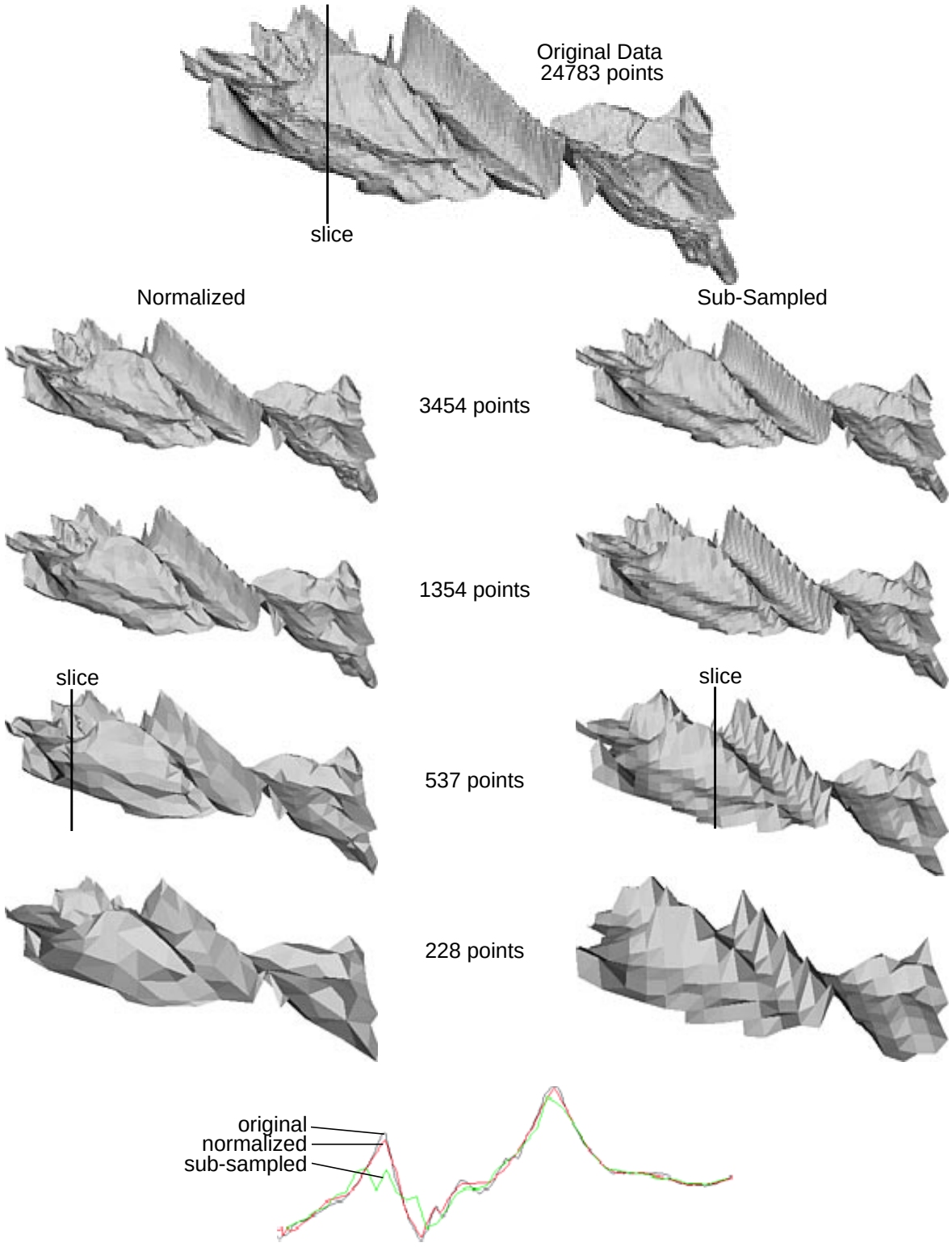


Figure A-15: Hierarchy of digital image elevation maps showing the advantage of simplification over sub-sampling. For example, the meshes generated using normalization accurately convey the prominent ridge through the hierarchy while the ridge in the sub-sampled hierarchy turns into peaks and valleys as the sub-sampling increases. Slicing through the third level of the hierarchy provides a 2-D confirmation of the benefit of normalization over sub-sampling.

Figure A-13 shows a hierarchy of normalized surface meshes generated from a range image of an industrial scene. The scene contains three I-beams, a pipe with two elbow joints and a water heater tank. The original data was generated from the range image by making each pixel in the range image a vertex and connecting pixels in adjacent rows and columns with edges. Range discontinuities were eliminated by removing extremely long edges from the surface mesh. Specular reflections off of a few surfaces in the scene cause incorrect range computations and result in holes in the original surface mesh. This results demonstrates the ability of our algorithm to handle meshes with significant mesh boundaries and holes. It also shows that, without modifying parameters, our algorithm can normalize meshes that contain both polyhedral (I-beams) and free-form (pipes) objects.

The final result demonstrates the use of our algorithm for simplifying digital elevation maps. The original surface mesh is generated from a digital elevation map of the Lockheed Martin facility in Denver, Colorado by making each pixel in the map a vertex and connecting pixels in adjacent rows and columns with edges. A mesh hierarchy generated with our algorithm is then shown next to a hierarchy of meshes generated through simple sub-sampling of the original digital elevation map. Each of the two meshes shown at each level of the hierarchy has the same number of points. It is clear from shaded views of the meshes that sub-sampling the mesh drastically changes the shape of the terrain map. For example, the prominent ridge in the terrain remains a ridge in the normalized hierarchy, while the ridge turns into a sequence of peaks and valleys in the sub-sampled hierarchy. The ability of the normalized hierarchy to preserve shape is also demonstrated in 2-D slices taken through the data for the third level of the hierarchy.

A.5 Conclusion

We have developed an algorithm that controls the resolution of a surface mesh by normalizing the lengths of the edges in the mesh while preserving mesh shape. The algorithm was developed with special attention to the types of surface meshes encountered in 3-D computer vision. It works equally well on meshes representing curved and polyhedral objects with or without boundaries. Our algorithm is similar to other mesh simplification algorithms in that it iteratively changes the mesh by applying local mesh operators, which in our case are edge-collapse and edge-split. Our algorithm differs from others in that the order in which edge operations are ap-

plied depends on the shape change induced in the mesh as well as on the length of the edge. It also uses a novel shape change measure that more accurately predicts the effect of applying a mesh operation. In particular, simplification along ridges is possible with our measure, while it is not possible with other shape change measures. Finally, our algorithm preserves the shape of the mesh during normalization by balancing expansion and shrinkage during edge-collapse and by applying a global bound on the maximum change in mesh shape.

In the future we plan to extend our algorithm in three directions. Instead of representing a mesh hierarchy using discrete levels, we would like to represent a hierarchy as a continuous stream of edge-collapse and edge-split operations. In this way, a mesh of arbitrary resolution could be generated from the stream by applying the operations in the stream until the desired resolution is reached. Representing the hierarchy as a stream of data would also allow gradual transmission of the model and continuous level of detail model generation. Another direction for this work is to explore the combination of geometry and texture or other surface properties in surface mesh normalization. The final direction we would like to explore is the introduction of topology changing operations into mesh normalization. Removing holes and merging surface patches could conceivably reduce the number of edges needed to describe a mesh at the desired resolution.

Appendix B

Spin-Image Similarity Measure

When spin-images are compared, only the bins where the two images have data are considered when calculating the correlation coefficient of the two images. This masking of pixels is done to account for occlusions in the scene data. As a side effect, the correlation coefficient will be calculated with different numbers of samples depending on the pair of spin-images that are being compared; the more the spin-images overlap, the more samples will be used in the calculation. It is intuitive that the more samples used in the calculation of correlation coefficient, the more confidence can be placed in the value calculated. Therefore, correlation coefficients calculated with many samples should be trusted more than correlation coefficients calculated with few samples. To incorporate this confidence when comparing spin-images, the variance of the correlation coefficient is added to the spin-image similarity measure. The mathematical derivation of our spin-image similarity measure is given below.

When two spin-images are compared, N samples $(x_1,y_1),...(x_N,y_N)$ are used to calculate the correlation coefficient R. The statistical distribution of R is complex. However, assuming that the samples come from a bivariate normal distribution, then the Fisher transformation [19] can be used to transform R into a random variable S

$$S = \operatorname{atanh}(R) = \frac{1}{2}\ln\left(\frac{1+R}{1-R}\right). \tag{B.1}$$

S has better statistical properties than R; namely, it comes from a normal distribution $N(\bar{S},\sigma_S)$ with well defined mean and variance

$$\bar{S} = \operatorname{atanh}(R) \qquad \sigma_S^2 = \frac{1}{N-3}. \tag{B.2}$$

Suppose we are trying to find the pair of spin-images that match the best. If all of the pairs of spin-images have the same number of samples *N*, an appropriate way to pick the best comparison would be to chose the pair *i* that maximizes correlation coefficient. This can be achieved by maximizing the expectation of S^2

$$\max_i \left(E[S_i^{\,2}]\right) \tag{B.3}$$

where *E[x]* is the expectation of *x*. However, if the pairs of spin-images have different numbers of samples, we need to construct a statistical loss function that measures correlation coefficient and confidence in correlation coefficient. Since confidence in correlation coefficient can be quantified by its variance, a feasible loss function that combines correlation coefficient and its variance, using the transformed correlation coefficient *S*, is

$$\max_i \left(E[S_i^{\,2} - \alpha\sigma_{S_i}^2]\right) = \max_i \left(E[S_i^{\,2}] - \alpha\sigma_{S_i}^2\right) \quad . \tag{B.4}$$

Since

$$\sigma_S^2 = E[S^2] - E[S]^2 \tag{B.5}$$

this reduces to choosing

$$\max_i \left(E[S_i]^2 - (1-\alpha)\sigma_{S_i}^2\right) = \max_i \left(\bar{S}_i^2 - (1-\alpha)\sigma_{S_i}^2\right) \quad . \tag{B.6}$$

Converting back to correlation coefficient and substituting $\lambda=(1-\alpha)$, the loss function becomes

$$C = \left(\operatorname{atanh}(R)^2 - \lambda\frac{1}{N-3}\right) . \tag{B.7}$$

Therefore, to find the best matching spin-images, we want to find the pair of spin-images that maximize the similarity measure defined by (B.7). In (B.7) the free variable λ weighs the importance of correlation coefficient against confidence in correlation coefficient. Our model dependent method for setting λ is described in Section 3.1.

Appendix C

Clutter Model

In object recognition, clutter is scene data that does not belong to the objects being searched for in the scene. Generally, clutter is caused by other objects located near the objects of interest. Since no a priori distinction can be made between clutter and objects of interest, clutter produces scene data that can confuse the object recognition system. Figure C-1 shows a model of a duck and then its placement in a cluttered scene.

In real scenes clutter is omnipresent, so any object recognition system designed for the real world must deal with clutter. Some systems perform segmentation before recognition to separate clutter from interesting object data. Other systems deal with clutter directly during the recognition process. Our algorithm falls in the latter class of recognition systems. To demonstrate how our system deals with clutter, we have developed a theoretical model of the effects of clutter on recognition using spin-images.

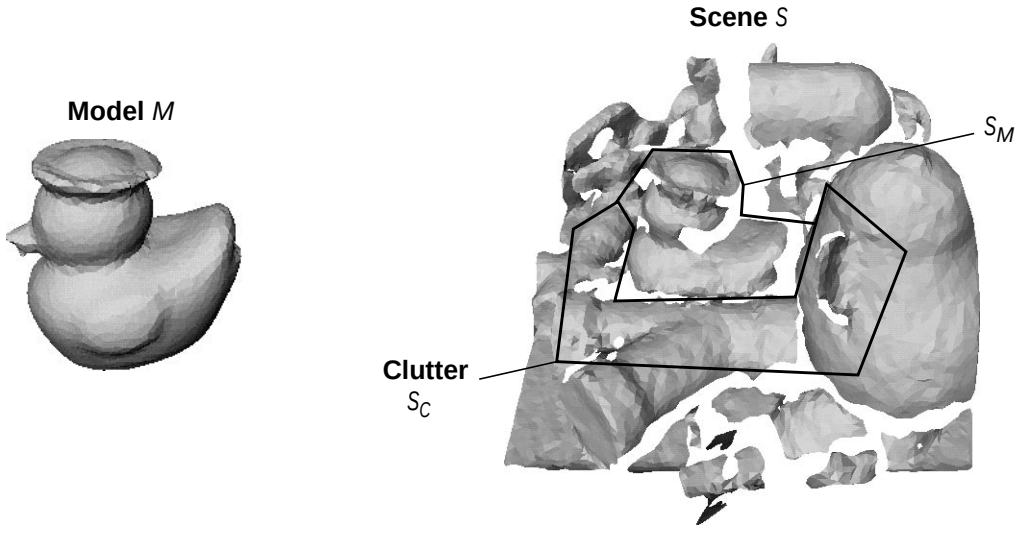


Figure C-1: Definition of clutter.

When clutter is present, the effects of clutter are manifested as a corruption of the pixel values of spin-images generated from the scene data. From our model, we conclude through reasoning about worst-cases, that spin-images are only moderately affected by clutter, and therefore can be used for recognition of objects in cluttered scenes. This conclusion stems from two theoretical results. First, when present, clutter is limited to connected regions in spin-images. Second, the worst case effect of clutter on correlation coefficient grows sub-linearly with the percent of clutter in a spin-image. The moderate effect of clutter on recognition is born out by results showing correct recognition of objects in the presence of cluttered scenes.

C.1 Geometric Model

The first step in understanding how clutter affects our recognition system is to develop a geometric model of how clutter affects the generation of spin-images. In other words, the pixels in a spin-image that are affected by the presence of clutter in the scene need to be determined. We will show that not all pixels can be affected by scene clutter because objects of non-zero thickness cannot intersect and the distance and angle constraints placed on points during spin-image generation prevent some clutter points from contributing to spin-image generation. Before continuing, some terminology must be established.

Suppose we have a complete 3-D model M and some scene data S that has two parts: S_M, a copy of the complete 3-D model M and S_C, some data that is not from M. Spin-images generated from points on M are not affected by clutter because the model is perfect. Spin-images generated form points on S_C are outliers, and are not investigated in our model of clutter. Therefore, we are concerned with finding the affect of clutter on the spin-image $I_{\boldsymbol{p},S}$ for a point $\boldsymbol{p}$ in the scene that is on S_M, the scene data representing the model. Stated formally, the fundamental question we would like to answer is: Given a model M and a scene S containing a point $\boldsymbol{p}$ on the representation of the model in the scene S_M, which pixels in $I_{\boldsymbol{p},S}$ can be affected by clutter S_C?

Since S_M is the same as M, only the pixels in $I_{\boldsymbol{p},S}$ that contain contributions from points on S_C will be corrupted. Therefore, we would like to determine which pixels in $I_{\boldsymbol{p},S}$ can possibly contain contributions from points on S_C. By finding all affected pixels, an upper bound on the number of cluttered pixels is determined. To make this problem tractable, some assumptions about the shape of the model and clutter need to be made.

To determine which pixels in $I_{p,S}$ can possibly contain contributions from points on S_C, we need to know the shape of the clutter. Instead of trying to model every shape that the clutter can possibly take, we will instead assume that the clutter comes from a single object of simple shape. To account for this gross simplification, we will allow the clutter object to occur in any feasible pose in the scene. This is reasonable because the clutter can appear anywhere in the scene. Therefore, when finding all pixels affected by clutter (the worst case effect of clutter), the clutter object should be placed in all feasible poses.

Suppose we would like to determine if a point $\boldsymbol{q}$ on S_C (in any pose) contributes to $I_{p,S}$ then there are three constraints on $\boldsymbol{q}$ that must be satisfied: the first is related to the non-zero thickness of objects and the next two are related to the constraints on points during spin-image generation.

Constraint 1: S_C and S_M cannot intersect because S_C and S_M come from separate objects.

Constraint 2: The distance between $\boldsymbol{p}$ and $\boldsymbol{q}$ must be less than the support distance D_s.

Constraint 3: The angle between the normals of $\boldsymbol{p}$ and $\boldsymbol{q}$ must be less than the support angle A_s.

If a point $\boldsymbol{q}$ satisfies the above constraints then the pixel that $\boldsymbol{q}$ is spin-mapped to will be cor-

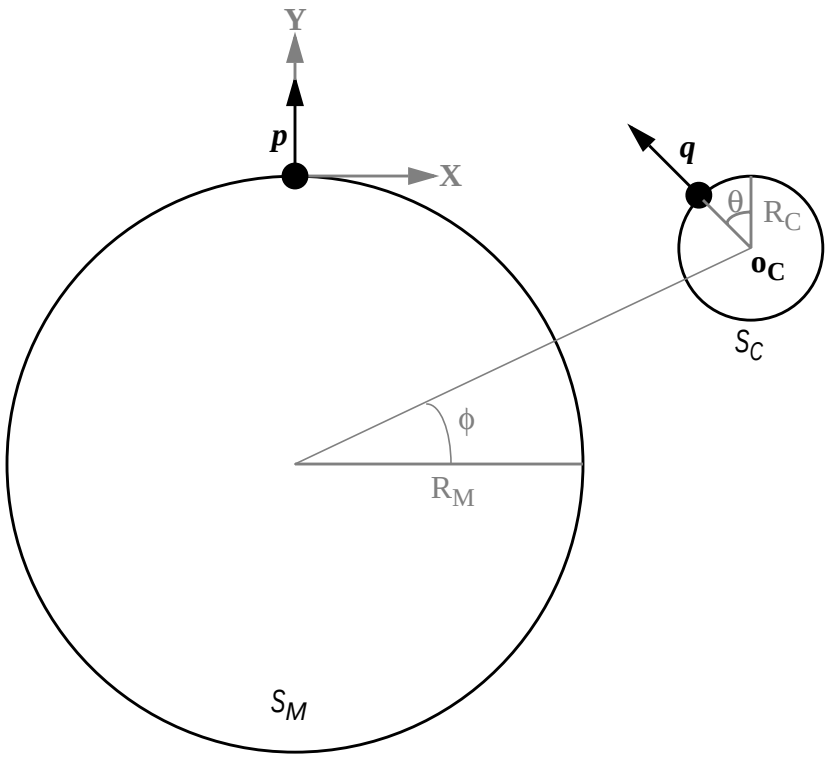


Figure C-2: Clutter geometry.

rupted. For each pixel, the goal is to determine if there exists a point ***q*** that spin-maps to the pixel and satisfies constraints 1 and 2 on S_C, in a pose satisfying constraint 1. Although constraints 2 and 3 are easy to check, determining if constraint 1 is satisfied requires knowledge of the shape of the objects.

The second assumption we make is that M (and consequently S_M) and S_C are spherical objects. This appears to be a gross simplification, but we will show that this model can be applied to objects of arbitrary shape. Let the radius of M (and S_M) be R_M and the radius of S_C be R_C. Now, satisfaction of the constraints can be determined through geometric arguments.

The constraints partition the space surrounding ***p*** into two regions: the region that can contain points satisfying the above constraints and its compliment. Because of the cylindrical symmetry of the spin-image generation process and the spherical symmetry of the objects involved, analysis of this partition of space can be reduced to the two dimensional problem of partioning a plane passing through ***p*** and parallel to the normal of ***p***. In the following 2-D construction, the origin is placed at ***p*** with the y-axis pointing along the normal of ***p*** as shown in Figure C-2. When S_C and S_M are touching, they are as close as they possibly can be and still satisfy constraint 1. This can be expressed as a circular constraint on the position of the origin $\mathbf{o_C} = (x_C, y_C)$ of S_C.

$$x_C^2 + (y_C + R_M)^2 \geq (R_M + R_C)^2 \tag{C.1}$$

The position of $\boldsymbol{q} = (x_q, y_q)$ on S_C with respect to the angle θ is

$$(x_q, y_q) = (x_C - R_C \sin\theta, y_C + R_C \cos\theta), \tag{C.2}$$

so the position of ***q*** is

$$(x_q + R_C \sin\theta)^2 + (y_q + R_M - R_C \cos\theta)^2 \geq (R_M + R_C)^2 . \tag{C.3}$$

Because S_C is a sphere, the normal of ***q*** has angle θ with respect to the y-axis, and the normal of ***p*** is along the y-axis. Therefore, constraint 3 can be rewritten as

$$\theta \leq A_s . \tag{C.4}$$

Combining (C.3) and (C.4) and the fact that the constraints are symmetric about the y-axis, we

get the following set of constraints on the position of $\boldsymbol{q}$

$$\text{if } \operatorname{atan}\left(\frac{|x_q|}{-(y_q + R_M)}\right) \geq A_s \text{ then } |x_q| + R_C \sin A_s)^2 + (y_q + R_M - R_C \cos A_s)^2 \geq (R_M + R_C)^2 \quad \text{(C.5)}$$

$$\text{if } \operatorname{atan}\left(\frac{|x_q|}{-(y_q + R_M)}\right) < A_s \text{ then } x_q{}^2 + (y_q + R_M)^2 \geq R_M{}^2 \quad \text{(C.6)}$$

Constraint 2 requires that the support of the $I_{\boldsymbol{p},S}$ is bounded and can be rewritten in terms of the coordinates of $\boldsymbol{q}$ as

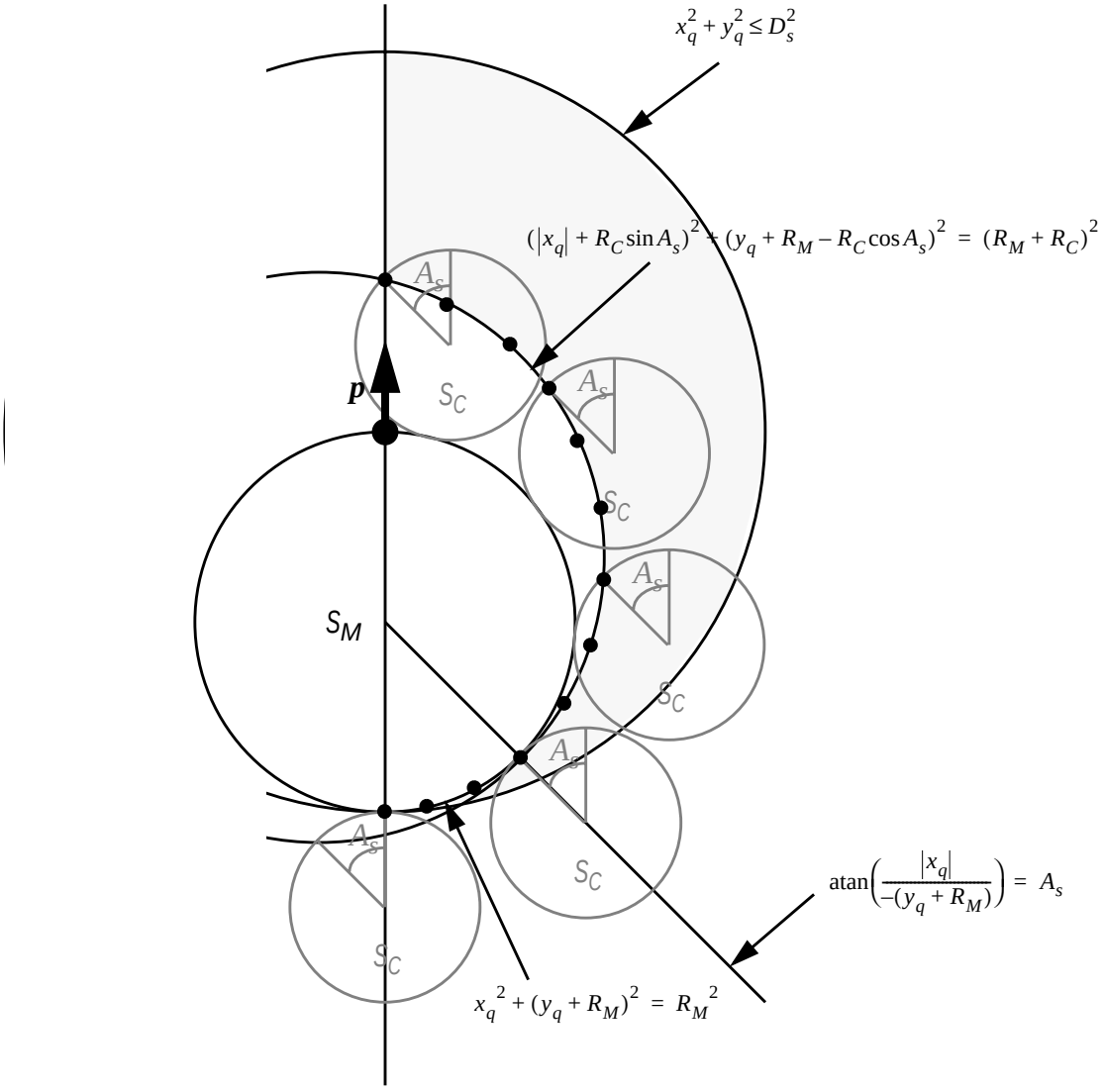


Figure C-3: The clutter region in the positive x plane for the parameters: $R_M = 1$, $R_C = 0.5$, $A_S = \pi/4$ and $D_s = 4$. The area of the spin-image support that can contain a clutter is shaded. The position of the clutter object is given for five different configurations on the border between the clutter and non-clutter regions.

$$x_q^2 + y_q^2 \leq D^2 \tag{C.7}$$

In spin-image generation, D_s actually sweeps out a cylindrical volume; the spherical constraint in (C.7) is a slight simplification. As shown in Figure C-3, these constraints partition the plane into a region affected by clutter (shaded) and a region not affected by clutter. In other words, the effect of clutter is contained within a connected region of the spin-image support. Our clutter model using spherical objects is a simplification, but it is sufficient to show that clutter can never affect the entire spin-image because the clutter and the model cannot intersect. This is a powerful result because it shows that some of the pixels in the spin-image (the ones not affected by clutter) will always contain good data.

An appropriate measure of the affect of clutter on spin-images is the ratio of the cluttered area over total spin-image area.

$$\text{Clutter Ratio} = \frac{\text{Clutter Area}}{\text{Total Area}} \tag{C.8}$$

The clutter measure varies from zero to one. Figure C-4 shows several plots that convey how the clutter measure varies depending on the choice of parameters (R_M, R_C, α, D_s) in the clutter model. The clutter measure in the plots was calculated numerically by sampling the plane on a regular grid and checking if the constraints were satisfied at each grid point. The number of points satisfying the constraints was then divided by the total number of points falling in the support of the spin-image. The general trends are that: as D_s increases, the clutter measure decreases; as A_s increases, the clutter measure decreases; and as R_C/R_M increases, the clutter measure decreases.

Our simplistic spherical model of clutter provides an upper bound on the total number of cluttered pixels possible in a spin-image. It assumes that, if a pixel can be cluttered, then the pixel will be cluttered. In practice this does not happen, because objects are rarely packed together this closely.

Our clutter model applies to objects of any shape if we make judicious choices for the model and clutter radii. An appropriate radius for the model object is the largest sphere contained entirely within the object that also intersects $\boldsymbol{p}$. This provides a lower bound on the possible spher-

ical size of the model. Since the area of clutter increases as the radius of the clutter decreases, the clutter radius should be as set as small as possible. The appropriate size for the spherical model of the clutter object is determined as follows. Select a point on the clutter object. Find

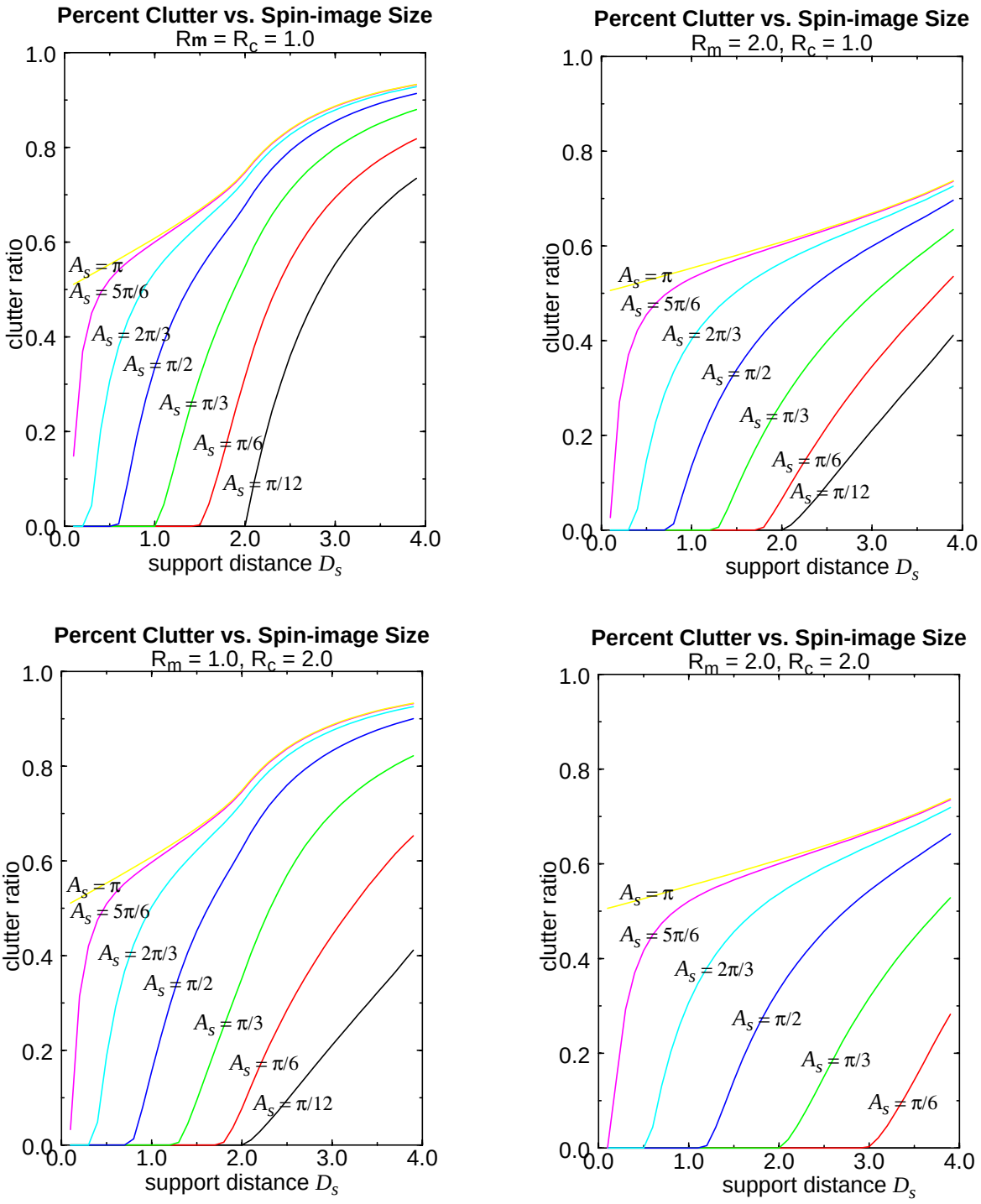


Figure C-4: Plots of clutter measure vs. spin-image support radius for different R_M and R_c. Each plot contains six plots for A_s = 0 to π (right to left).

the largest sphere that touches the point and is still contained within the clutter object. Take the minimum diameter of these spheres over all points on the clutter object and you have the appropriate radius. By setting the model and clutter radii to these values, the spherical clutter model can be used to find a lower bound on the clutter measure. In the next section, we will investigate the effect of clutter on the correlation coefficient of two spin-images.

C.2 Effect of Clutter On Correlation Coefficient

Central to our recognition system is the comparison of spin-images by calculation of the correlation coefficient. Therefore, to truly determine the effect of clutter on recognition, we need to determine the effect of clutter on the calculation of correlation coefficient. We determined in the previous section that clutter affects only a portion of any scene spin-image. What we would like to determine is the worst case effect of clutter on correlation coefficient given a certain level of clutter in the spin-images.

Our model of the effect of clutter on correlation coefficient is as described in the previous section. There is a complete 3-D model M and some scene data S that has two parts: S_M, a copy of the complete 3-D model M and S_C, some data that is not from M. Suppose that on the model there is a point $\boldsymbol{m}$ whose correct corresponding point in the scene is $\boldsymbol{s}$. We would like to determine the effect of clutter on the correlation coefficient between the spin-images of the two points, $I_{\boldsymbol{s},S}$ (with pixels x_i^s) and $I_{\boldsymbol{m},M}$ (with pixels x_i^m) given a particular amount of clutter in the scene. Let the total number of pixels in the spin-images be N and the total number of pixels in the image that can be corrupted by clutter be $N_C<N$. Let the subset of $I_{\boldsymbol{s},S}$ that is corrupted by clutter be I_C and the portion that is not corrupted be I_N. Since the scene contains a complete copy of the model, the pixels of $I_{\boldsymbol{C}}$ will be greater than the pixels of $I_{\boldsymbol{m},M}$, and the pixels of I_N will be equal to the pixels of $I_{\boldsymbol{m},M}$. Clutter is caused by extraneous scene points that are not part of the model, and the number of these points is bounded. Therefore, it is reasonable to assume that the total contribution of clutter to any cluttered pixel in the scene spin-image is bounded. Suppose the bound on the addition to the pixels is $0 \le \delta_i \le \delta$, then

$$\text{if } x_i^s \in I_N \text{ then } x_i^s = x_i^m \tag{C.9}$$

$$\text{if } x_i^s \in I_C \text{ then } x_i^s = x_i^m + \delta_i \tag{C.10}$$

We can also assume without loss of generality that the model pixels have been normalized such that $x_i^m \le 1$, which implies that

$$\overline{x_m} = \frac{1}{N}\sum_{I_{m,M}} x_i^m \le 1 \text{ and } \sigma_m^2 = \frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 - \left(\frac{1}{N}\sum_{I_{m,M}} x_i^m\right)^2 \le 1 \tag{C.11}$$

The correlation coefficient ρ is defined as

$$\frac{\frac{1}{N}\sum x_i^m x_i^s - \frac{1}{N^2}\sum x_i^m \sum x_i^s}{\left(\left(\frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 - \left(\frac{1}{N}\sum_{I_{m,M}} x_i^m\right)^2\right)\left(\frac{1}{N}\sum_{I_{s,S}} (x_i^s)^2 - \left(\frac{1}{N}\sum_{I_{s,S}} x_i^s\right)^2\right)\right)^{\frac{1}{2}}} \tag{C.12}$$

To determine the worst case effect of clutter on ρ, we need to determine the smallest possible numerator value and the largest possible denominator value that are independent of specific pixel values. For the numerator, we have

$$\frac{1}{N}\sum x_i^m x_i^s - \frac{1}{N^2}\sum x_i^m \sum x_i^s$$

$$= \frac{1}{N}\sum_{I_N} x_i^m x_i^m + \frac{1}{N}\sum_{I_C} x_i^m (x_i^m + \delta_i) - \frac{1}{N^2}\sum_{I_{m,M}} x_i^m \left(\sum_{I_N} x_i^m + \sum_{I_C} (x_i^m + \delta_i)\right)$$

$$= \frac{1}{N}\sum_{I_N} x_i^m x_i^m + \frac{1}{N}\sum_{I_C} x_i^m x_i^m + \frac{1}{N}\sum_{I_C} x_i^m \delta_i - \frac{1}{N^2}\sum_{I_{m,M}} x_i^m \left(\sum_{I_N} x_i^m + \sum_{I_C} x_i^m + \sum_{I_C} \delta_i\right)$$

$$= \frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 + \frac{1}{N}\sum_{I_C} x_i^m \delta_i - \frac{1}{N^2}\sum_{I_{m,M}} x_i^m \left(\sum_{I_{m,M}} x_i^m + \sum_{I_C} \delta_i\right)$$

$$= \frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 + \frac{1}{N}\sum_{I_C} x_i^m \delta_i - \frac{1}{N^2}\left(\sum_{I_{m,M}} x_i^m\right)^2 - \frac{1}{N^2}\left(\sum_{I_{m,M}} x_i^m\right)\left(\sum_{I_C} \delta_i\right)$$

$$= \frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 - \frac{1}{N^2}\left(\sum_{I_{m,M}} x_i^m\right)^2 + \frac{1}{N}\sum_{I_C} x_i^m \delta_i - \frac{1}{N^2}\left(\sum_{I_{m,M}} x_i^m\right)\left(\sum_{I_C} \delta_i\right)$$

$$= \sigma_m^2 + \frac{1}{N}\sum_{I_C} x_i^m \delta_i - \frac{1}{N}\sum_{I_C} \overline{x_m}\delta_i$$

$$= \sigma_m^2 + \frac{1}{N}\sum_{I_C} (x_i^m - \overline{x_m})\delta_i$$

$$\geq \sigma_m^2 + \frac{1}{N}\sum_{I_C} (-1)\delta_i \text{ because } x_i^m \leq 1$$

$$\geq \sigma_m^2 - \frac{1}{N}\sum_{I_C} \delta \text{ because } 0 \leq \delta_i \leq \delta, \text{ so}$$

$$= \sigma_m^2 - \frac{N_C}{N}\delta \qquad \text{(C.13)}$$

For the denominator, we have

$$= \left(\left(\frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 - \left(\frac{1}{N}\sum_{I_{m,M}} x_i^m\right)^2\right)\left(\frac{1}{N}\sum_{I_{s,S}} (x_i^s)^2 - \left(\frac{1}{N}\sum_{I_{s,S}} x_i^s\right)^2\right)\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\frac{1}{N}\left(\sum_{I_N} (x_i^m)^2 + \sum_{I_C} (x_i^m + \delta_i)^2\right) - \left(\frac{1}{N}\left(\sum_{I_N} x_i^m + \sum_{I_C} (x_i^m + \delta_i)\right)\right)^2\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\frac{1}{N}\left(\sum_{I_N} (x_i^m)^2 + \sum_{I_C} ((x_i^m)^2 + 2x_i^m\delta_i + \delta_i^2)\right) - \left(\frac{1}{N}\left(\sum_{I_N} x_i^m + \sum_{I_C} x_i^m + \sum_{I_C} \delta_i\right)\right)^2\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 + \frac{1}{N}\sum_{I_C} (2x_i^m\delta_i + \delta_i^2) - \left(\frac{1}{N}\left(\sum_{I_{m,M}} x_i^m + \sum_{I_C} \delta_i\right)\right)^2\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 + \frac{1}{N}\sum_{I_C} (2x_i^m\delta_i + \delta_i^2) - \frac{1}{N^2}\left(\sum_{I_{m,M}} x_i^m\right)^2 - \frac{2}{N^2}\sum_{I_{m,M}} x_i^m \sum_{I_C} \delta_i - \frac{1}{N^2}\left(\sum_{I_C} \delta_i\right)^2\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\frac{1}{N}\sum_{I_{m,M}} (x_i^m)^2 - \frac{1}{N^2}\left(\sum_{I_{m,M}} x_i^m\right)^2 + \frac{1}{N}\sum_{I_C} (2x_i^m\delta_i + \delta_i^2) - \frac{2}{N}\overline{x_m}\sum_{I_C} \delta_i - \frac{1}{N^2}\left(\sum_{I_C} \delta_i\right)^2\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\sigma_m^2 + \frac{1}{N}\sum_{I_C}(2x_i^m\delta_i + \delta_i^2) - \frac{2}{N}\overline{x_m}\sum_{I_C}\delta_i - \frac{1}{N^2}\left(\sum_{I_C}\delta_i\right)^2\right)^{\frac{1}{2}}$$

$$= \sigma_m\left(\sigma_m^2 + \frac{1}{N}\sum_{I_C}(2(x_i^m - \overline{x_m})\delta_i + \delta_i^2) - \frac{1}{N^2}\left(\sum_{I_C}\delta_i\right)^2\right)^{\frac{1}{2}}$$

$$\leq \sigma_m\left(\sigma_m^2 + \frac{1}{N}\sum_{I_C}(2\delta_i + \delta_i^2) - \frac{1}{N^2}\left(\sum_{I_C}\delta_i\right)^2\right)^{\frac{1}{2}} \text{ because } x_i^m - \overline{x_m} \leq 1$$

$$\leq \sigma_m\left(\sigma_m^2 + \frac{1}{N}\sum_{I_C}(2\delta_i + \delta_i^2)\right)^{\frac{1}{2}} \text{ by dropping negative term}$$

$$\leq \sigma_m\left(\sigma_m^2 + \frac{N_C}{N}(2\delta + \delta^2)\right)^{\frac{1}{2}} \text{ because } 0 \leq \delta_i \leq \delta. \qquad \text{(C.14)}$$

Combining (C.13) and (C.14), we get the lower bound on the correlation coefficient given that the variance of the model data σ_m^2, the clutter in each pixel is bounded by δ, and the number of clutter pixels is bounded by N_C.

$$\rho_{LB} = \frac{\sigma_m^2 - \frac{N_C}{N}\delta}{\sigma_m\left(\sigma_m^2 + \frac{N_C}{N}(2\delta + \delta^2)\right)^{\frac{1}{2}}} \qquad \text{(C.15)}$$

Plotting (C.15)for different values of N_C/N, δ, and σ_m^2 will convey the dependence of the lower bound on ρ for different clutter situations. By construction, $N_C/N<1$ and $\sigma_m^2 < 1$. It is also feasible (although not guaranteed) that $\delta<1$ because the contribution of clutter to the spin-image pixels will generally not be greater than the normalized model pixels $x_i^m \leq 1$. Figure C-5 shows some plots of ρ_{LB} for different parameter values. From the plots, it is clear that ρ_{LB} decreases sub-linearly with the percent of clutter in the scene N_C/N.

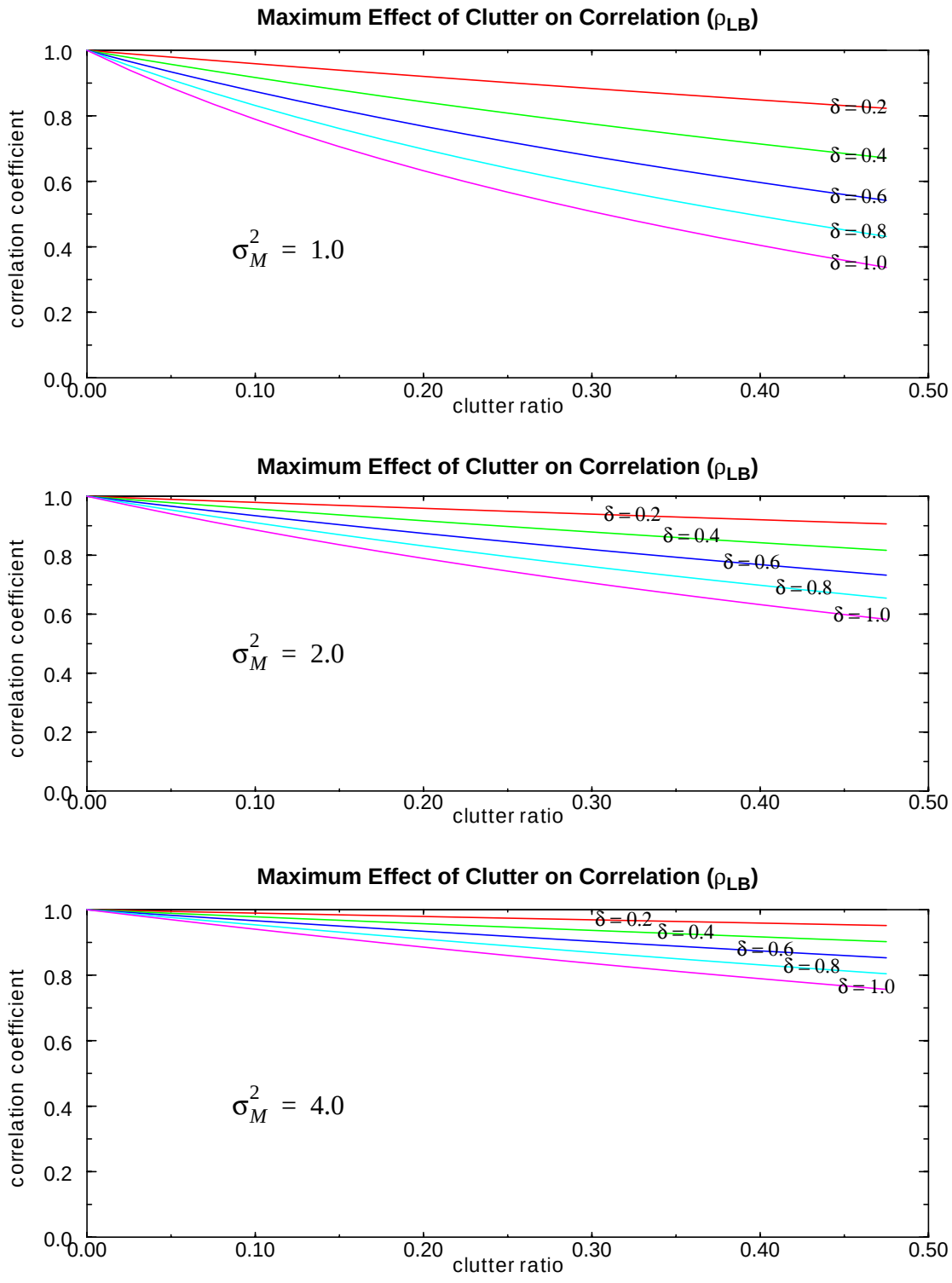


Figure C-5: Plots of the lower bound on correlation coefficient.

Bibliography

[1] F. Arman and J. K. Aggarwal. CAD-based vision: object recognition in cluttered range images using recognition strategies. *Computer Vision, Graphics and Image Processing*, vol. 58, no. 1, pp. 33-48, 1993.

[2] C. Bajaj, F. Bernardini and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3-D scans. *Proc. Computer Graphics (SIGGRAPH '95)*, pp. 109-118, August 1995.

[3] R.E. Barry, C. Little, and B. Burks. Requirements and Design Concept for a Facility Mapping System. *Proc. ANS 6th Topical Meeting on Robotics and Remote Systems (ANS '95)*, pp. 775-783, February 1995.

[4] P. Belhumeur, J. Hespanha and D. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *Proc. European Conference on Computer Vision (ECCV '96)*, 1996.

[5] J. Bentley. Multidimensional binary search trees used for associative learning. *Communications of the Association for Computing Machinery*, vol. 18, no. 9, pp. 509-517.

[6] R. Bergevin, D. Laurendeau and D. Poussart, Registering range views of multipart objects, *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 1-16, 1995.

[7] P. Besl and N. McKay. A method of registration of 3-D shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 239-256, February 1992.

[8] P. Besl. The triangle as primary representation. *Object Representation in Computer Vision*, M. Hebert, J . Ponce, T. Boult and A. Gross (Eds.), Springer Verlag, Berlin, 1995.

[9] F. Betting, J. Feldmar, N. Ayache and F. Dervernay. A new framework for fusing stereo images with volumetric medical images. *Proc. First International Conference on Computer Vision, Virtual Reality an Robotics in Medicine (CVRMed '95)*, pp. 30-39, 1995.

[10] J. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. on Graphics*, vol. 3, no. 4, pp. 266-286, October 1984.

[11] T. Cass. Polynomial-time geometric matching for object recognition. *Int'l Jour. Computer Vision*, vol. 21(1/2), pp. 37-61, 1997.

[12] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, vol. 10, no. 3, pp. 145-155, 1992.

[13] Y. Chen and G. Medioni. Surface description of complex objects from range images. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '94)*, pp. 153-158, 1994.

[14] C. Chua and R. Jarvis. 3-D free-form surface registration and object recognition. *Int'l Jour. Computer Vision*, vol. 17, no. 1, pp. 77-99, 1996.

[15] L. Conley, W. Hamel and B. Thompson. Rosie: A mobile worksystem for decontamination and dismantlement operations. *Proc. ANS 6th Topical Meeting on Robotics and Remote Systems (ANS '95)*, pp. 231-238, 1995.

[16] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Proc. Computer Graphics (SIGGRAPH '96)*, August 1996.

[17] H. Dellingette, M. Hebert and K. Ikeuchi. A spherical representation for the recognition of curved objects. *Proc. Computer Vision and Pattern Recognition (CVPR '93)*, pp. 103-112, 1993.

[18] H. Dellingette, M. Hebert and K. Ikeuchi. Shape representation and image segmentation using deformable surfaces. *Image and Vision Computing*, vol. 10, no. 3, pp. 132-144, 1992.

[19] J. Devore. *Probability and Statistics for Engineering and Sciences*. Brooks/Cole, Belmont, CA, 1987.

[20] D. Dion Jr., D. Laurendeau and R. Bergevin. Generalized cylinder extraction in range images. *Proc. Int'l Conf. on Recent Advance in 3-D Digital Imaging and Modeling*, Ottawa, pp. 141-147, May 1997.

[21] C. Dorai and A. Jain. COSMOS - A representation scheme for 3D free-form objects. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp 1115-1130, 1997.

[22] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York, 1973.

[23] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for images analysis. *Jour. Mathematical Imaging and Vision*, vol. 4, no. 4, pp. 353-373, December 1994.

[24] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle. Multiresolution analysis of arbitrary meshes. *Proc. Computer Graphics (SIGGRAPH '95)*, pp. 173-182, 1995.

[25] A. Elfies. Sonar-based real world mapping and navigation. *IEEE Jour. Robotics and Automation*, vol. RA-3, no. 3, pp. 249-265, 1987.

[26] O. Faugeras and M. Hebert. The representation, recognition and locating of 3-D objects. *Int'l. Jour. Robotics Research*, vol. 5, no. 3, pp. 27-52, 1986.

[27] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, Cambridge, MA, 1993.

[28] R. Fisher. Representation, extraction and recognition with second-order topographic surface features. *Image and Vision Computing*, vol. 10, no. 3, pp. 156-168, 1992.

[29] P. Flynn and A. Jain. BONSAI: 3D object recognition using constrained search. vol. 13, no. 10, pp. 1066-1075, October 1991.

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

[31] J. Friedman, J. Bentley and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. on Mathematical Software*, vol. 3, no. 3, pp. 209-226, September 1977.

[32] K. Fukanaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[33] W.E.L. Grimson. Localizing overlapping parts by searching the interpretation tree. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp 469-482, 1987.

[34] W.E.L. Grimson. *Object Recognition by Computer: The Role of Geometric Constraints*. MIT Press, Cambridge, MA, 1990.

[35] A. Guéziec. Surface simplification with variable tolerance. *Proc. Medical Robotics and Computer Assisted Surgery (MRCAS '95)*, pp. 132-139, November, 1995.

[36] A. Guéziec and N. Ayache. Smoothing and matching of 3-D space curves. *Int'l Jour. Computer Vision*, vol. 12, no. 1, pp. 79-104, 1994.

[37] M. Hebert, J. Ponce, T. Boult and A. Gross (Eds.). *Object Representation in Computer Vision*. Springer-Verlag, Berlin, 1995.

[38] M. Hebert, K. Ikeuchi and H. Delingette. A spherical representation for recognition of free-form surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 681-689, 1995.

[39] P. Heckbert and M. Garland. Survey of polygonal surface simplification algorithms. Technical Report CMU-CS-97-TBD, The School of Computer Science, Carnegie Mellon University, 1997.

[40] Y. Hecker and R. Bolle. On geometric hashing and the generalized hough transform. *IEEE Trans. Systems, Man and Cybernetics*, vol. 24, no. 9, pp. 1328-1338, 1994.

[41] A Hilton, A. Stoddart, J. Illingworth and T Windeatt. Reliable surface reconstruction from multiple range images. *Fourth European Conf. on Computer Vision (ECCV `96)*, pp. 14-18, April 1996.

[42] P. Hinker and C. Hansen. Geometric optimization. *Proc. Visualization '93*, pp. 189-195, October 1993.

[43] H. Hoppe, T. DeRose, T. DuChamp, J. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. *Proc. Computer Graphics (SIGGRAPH '92)*, pp. 71-78. July 1992.

[44] H. Hoppe. Progressive Meshes. *Proc. Computer Graphics (SIGGRAPH '96)*, pp. 99-108, 1996.

[45] B. Horn. Closed-form solution of absolute orientation using unit quaternions. *Jour. Optical Society of America*, vol. 4, no. 4, pp. 629-642, 1987.

[46] D. P. Huttenlocher and S. Ullman. Recognizing Solid objects by alignment with an image. *Int'l Jour. Computer Vision*, vol. 5, no. 2, pp. 195-212, 1990.

[47] K. Ikeuchi, T. Shakunaga, M. Wheeler and T. Yamazaki, Invariant Histograms and deformable template matching for SAR target recognition, *Proc. Computer Vision and Pattern Recognition (CVPR '96)*, pp. 100-105, 1996.

[48] A. Johnson and M. Hebert. Surface registration by matching oriented points. *Proc. Int'l Conf. on Recent Advance in 3-D Digital Imaging and Modeling*, pp. 121-128, Ottawa, May 1997.

[49] A. Johnson and M. Hebert. Control of mesh resolution for 3-D object recognition, Technical Report CMU-RI-TR-96-20, The Robotics Institute, Carnegie Mellon University, October 1996.

[50] A. Johnson and M. Hebert.Object recognition by matching oriented points. *Proc. Computer Vision and Pattern Recognition (CVPR '97)*, pp. 684-689, San Juan, P.R, May 1997.

[51] A. Johnson, P. Leger, R. Hoffman, M. Hebert, and J. Osborn. 3-D object modeling and recognition for telerobotic manipulation. *Proc. Intelligent Robots and Systems (IROS '95)*, pp. 103-110, 1995.

[52] A. Johnson, R. Hoffman, J. Osborn, and M. Hebert. A system for semi-automatically modeling of complex environments. *Proc. Int'l Conf. on Recent Advance in 3-D Digital Imaging and Modeling (3DIM '97)*, Ottawa, pp. 213-220, May 1997.

[53] A. Johnson and S. Kang. Registration and integration of textured 3-D data. *Proc. Int'l Conf. on Recent Advance in 3-D Digital Imaging and Modeling (3DIM '97)*, Ottawa, pp. 234-241, 1997.

[54] A. Kalvin and R. Taylor. Superfaces: polyhedral approximation with bounded error. *SPIE Medical Imaging*, vol. 2164, pp. 2-13, 1994.

[55] S. Kang, A. Johnson and R. Szeliski. Extraction of concise and realistic 3-D models from real data. Technical Report CRL 95/7, Digital Equipment Corporation Cambridge Research Lab., October 1995.

[56] V. Koivunen and R. Bajcsy. Spline Representations in 3-D Vision, in *Object Representation in Computer Vision*, M. Hebert, J. Ponce, T. Boult and A. Gross, (Eds.) Springer-Verlag, pp. 177-190, December 1994.

[57] C. Kolb. Rayshade User Guide and Reference Manual. August 1994.

[58] I. S. Kweon, R. Hoffman, and E. Krotkov. Experimental Characterization of the Perceptron Laser Rangefinder. Technical Report CMU-RI-TR-91-1, The Robotics Institute, Carnegie Mellon University, January 1991.

[59] Y. Lamdan and H. Wolfson. Geometric Hashing: a general and efficient model-based recognition scheme. *Proc. Second Int'l Conf. Computer Vision (ICCV '88)*, pp. 238-249, 1988.

[60] Y. Lamdan and H. Wolfson. On the error analysis of 'Geometric Hashing.' *Proc. Computer Vision and Pattern Recognition 1991 (CVPR '91)*, pp. 22-27, 1991.

[61] W. Lorensen and H. Cline. Marching Cubes: a high resolution 3D surface construction algorithm. *Proc. Computer Graphics (SIGGRAPH '87)*, 163-169, 1987.

[62] M. Martin and H. Moravec. Robot Evidence Grids. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carnegie Mellon University, March 1996.

[63] L. Matthies and S. Shafer. Error modeling in stereo navigation. *IEEE Jour. Robotics and Automation*, vol. RA-3, no. 3, pp. 239-248, June 1987.

[64] C. Montani, R. Scateni and R. Scopigno. A modified look-up table for implicit disambiguation of Marching Cubes. *Visual Computer*, vol. 10, pp. 353-355, 1994.

[65] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int'l Jour. Computer Vision*, vol. 14, pp. 5-24, 1995.

[66] S. Näher and Christian Urhig. *The LEDA User Manual: Version R 3.3*. Max-Planck-Institut für Informatik, 1996.

[67] S. Nene and S. Nayar. A simple algorithm for closest point search in high dimensions. Technical Report CUCS-030-95, Columbia University, October 1995.

[68] A. Pentland and S. Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 7, pp. 715-729, 1991.

[69] F. Pipitone and W. Adams. Tripod operators for recognizing objects in range images; rapid rejection of library objects. *Proc. IEEE Robotics and Automation (R&A 1992)*, pp. 1596-1601, 1992.

[70] F. Preparata and M. Shamos. *Computational Geometry: An Introduction.* Springer-Verlag, New York, 1985.

[71] W. Press, S. Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition.* Cambridge University Press, Cambridge, UK, 1992.

[72] N. Raja and A. Jain. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, vol. 10, no. 3, pp. 179-190, 1992.

[73] I. Rigoutsos and R. Hummel. A Bayesian approach to model matching with geometric hashing. *Computer Vision and Image Understanding*, vol. 62, no. 1, pp. 11-26, July 1995.

[74] M. Rutishauser, M. Stricker and M. Trobina. Merging range images of arbitrarily shaped objects. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '94)*, pp. 573-580, 1994.

[75] Y. Sato and K. Ikeuchi. Reflectance analysis for 3D computer graphics model generation. *Graphical Models and Image Processing*, Vol. 58, No. 5, pp. 437-451, 1996.

[76] W. Schroeder, J. Zarge and W. Lorensen. Decimation of triangular meshes. *Proc. Computer Graphics (SIGGRAPH '92)*, pp. 65-70, 1992.

[77] S. Sclaroff and A. Pentland. Model matching for correspondence and recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence.* vol. 17, no. 6, pp. 545-561, 1995.

[78] W. Schroeder. A global error bound for triangle decimation. *WWW download, http://www.crd.ge.com/~schroeder, 1996.*

[79] K. Shimada. Physically-based automatic mesh generation. *Jour. Japan Society for Simulation Technology*, vol. 12, no. 1, pp. 11-20, 1993.

[80] H. Y. Shum, K. Ikeuchi and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. Pattern Analysis and Machine Intelligence*. vol. 17, no. 9, 1995.

[81] D. Simon, M. Hebert and T. Kanade. Real-time 3-D pose estimation using a high-speed range sensor. *Proc. Int'l Conf. Robotics and Automation (R&A '94)*, May 1994.

[82] D. Simon. *Fast and Accurate Shape-Based Registration*. Ph.D. Thesis, The Robotics Institute, Carnegie Mellon University, November 1996.

[83] M. Soucy and D. Laurendeau. Multi-resolution surface modeling from multiple range views. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '92)*, pp. 348-353, 1992.

[84] R. Szeliski, D. Tonnensen and D. Terzopoulos. Modeling surfaces of arbitrary topology using dynamic particles. *Proc. Computer Vision and Pattern Recognition (CVPR '93)*, pp. 92-87, 1993.

[85] R. Sproull. Refinements to nearest neighbor searching in k-dimensional trees. *Algorithmica* vol. 6, pp. 579-589, 1991.

[86] F. Stein and G. Medioni. Structural Indexing: efficient 3-D object recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 125-145, 1992.

[87] D. Terzopoulos and M. Vasilescu. Sampling and reconstruction with adaptive meshes. *Proc. Computer Vision and Pattern Recognition (CVPR '91)*, pp. 70 - 75, 1991.

[88] G. Taubin . A Signal processing approach to fair surface design. *Proc. Computer Graphics (SIGGRAPH '95)*, pp. 351-358, 1995.

[89] G. Taubin. Discrete surface signal processing: the polygon as the surface element. *Object Representation in Computer Vision*, M. Hebert, J . Ponce, T. Boult and A. Gross (Eds.), Springer Verlag, Berlin, 1995.

[90] S. Thayer, S., C. Gourley, M. Trivedi, C. Chen, S. Marapane, P. Butler and H Costello. On-line stereo vision and graphical interface for decontamination and decommissioning applications using the advanced servo manipulator. *Proc. ANS 5th Topical Meeting on Robotics and Remote Systems (ANS '95)*, pp. 287-294, April 1993.

[91] J. Thirion. New feature points based on geometric invariants for 3D image registration. *Int'l Jour. Computer Vision*, vol. 18, no. 2, pp. 121-137, 1996.

[92] G. Turk. Re-tiling polygonal surfaces. *Proc. Computer Graphics (SIGGRAPH '92)*, pp. 55-64, 1992.

[93] G. Turk and M. Levoy. Zippered polygonal meshes from range images. *Proc. Computer Graphics (SIGGRAPH '94)*, pp. 311-318, 1994.

[94] M. Turk and A. Pentland. Face Recognition using eigenfaces. *Proc. Computer Vision and Pattern Recognition (CVPR '91)*, pp 586-591, 1991.

[95] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 13, no. 10, pp. 992-1006, 1991.

[96] R. Veltkamp. 2D and 3D object reconstruction with the γ-neighborhood graph. Technical Report CS-R9116, CWI Centre for Mathematics and Computer Science, 1991.

[97] M. Wheeler. *Automatic Modeling and Localization for Object Recogntion*. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, October 1996.

[98] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. *Proc. Computer Graphics (SIGGRAPH '94)*, pp. 269-277, July 1994.

[99] D. Zhang and M. Hebert. Multi-scale classification of 3-D objects. *Proc. Computer Vision and Pattern Recognition (CVPR '97)*, pp 864-869, 1997.

[100] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int'l Jour. Computer Vision*, vol. 13, no. 2, pp. 119-152, 1994.