Recognizing Objects by Matching Oriented Points

Andrew E. Johnson and Martial Hebert

CMU-RI-TR-96-04

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

May 1996

This research was supported by the Department of Energy under contract DE-AC21-92MC29104.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. government.

Abstract

By combining techniques from geometric hashing and structural indexing, we have developed a new representation for recognition of free-form objects from three dimensional data. The representation comprises descriptive spin-images associated with each oriented point on the surface of an object. Constructed using single point bases, spin-images are data level shape descriptions that are used for efficient matching of oriented points. During recognition, scene spin-images are indexed into a stack of model spin-images to establish point correspondences between a model object and scene data. Given oriented point correspondences, a rigid transformation that maps the model into the scene is calculated and then refined and verified using a modified iterative closest point algorithm.

Indexing of oriented points bridges the gap between recognition by global properties and feature based recognition without resorting to error-prone segmentation or feature extraction. It requires no knowledge of the initial transformation between model and scene, and it can register fully 3-D data sets as well as recognize objects from partial views with occlusions. We present results showing simultaneous recognition of multiple 3-D anatomical models in range images and range image registration in the context of interior modeling of an industrial facility.

KEYWORDS: 3-D object recognition, oriented point, structural indexing, range image registration, free-form surface, polygonal surface mesh, 3-D modeling.

Table of Contents

List of Figures

1 Introduction

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. In robotics, 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, so, 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, and 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, thus 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, etc.) 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, has the following stages. First a representation of the shape 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 parts of the stored model representation that are similar to parts of the processed 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. The object recognition system presented in this paper follows the recognition from localization paradigm using data from three dimensional sensors.

Three-dimensional object recognition uses the true 3-D shape of objects in its model representation. Commonly used sensors in robotics for generating 3-D data are multiple camera stereo rigs, laser rangefinders and structured light range cameras. Three dimensional data can also be acquired with sonar, tactile, computed tomography and magnetic resonance sensors. 3-D data can come in the form of depth maps, isolated 3-D points and lines, or 3-D intensity images, depending on the sensor and sensing algorithm. We use polygonal surface meshes for our 3-D representation because surface meshes can represent objects of arbitrary shape and topology and the data from most 3-D sensors used in robotics can be converted to a surface mesh.

Another source of variation among 3-D model-based object recognition systems is the means by which 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.

In this paper we present a new model representation for three dimensional object recognition that combines the descriptive nature of global object properties with the robustness to partial views and clutter of local features. Specifically, oriented points are used to create a mapping of 3-D points to a 2-D image space at each vertex of a surface mesh. By applying the unique mapping at each point in the surface mesh to all other points in the surface mesh, a spin-image is generated that encodes the global shape of the object with respect to the current point. At recognition time, spin-images from points on the model are compared to spin-images from points in the scene; when two spin-images are similar enough, a point correspondence between model and scene is established. Several point correspondences can then be used to calculate a transformation from model to scene for verification. Since a spin-image is generated at each point in the surface mesh using all other points on the object, error prone feature extraction and segmentation are avoided. To verify matches between model and scene, we have developed an efficient iterative closest point verification algorithm which verifies matches and refines transformations from model to scene. Used in conjunction with our correspondence algorithm, the verification algorithm provides accurate and efficient recognition. Since we make no assumptions about the topology or shape of the objects represented, our algorithm can recognize arbitrarily shaped objects; and, since no initial transformation is required, our algorithm can be used for object recognition in completely unknown environments.

1.1 Related Work

A comprehensive overview of model representation in 3-D object recognition is given in [9] and comprehensive overviews of methods for establishing correspondences and verifying matches are given in [6][7]. Because of their direct relevance to our algorithm, some review of work in the areas of geometric hashing, structural indexing and the iterative closest point algorithms is necessary.

The term geometric hashing was first coined by Lamdan and Wolfson [13][14] to describe their work utilizing a look up table to speed the establishment of correspondences in recognition. Geo-

metric hashing can be broken into two categories: basis geometric hashing and structural indexing or curve based geometric hashing [10]. 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 features.

Structural indexing or curve geometric hashing does not use tuples of features like basis geometric hashing, but 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 [17]. In the preprocessing stage, signatures are computed for each model and 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 [8] 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 [2] 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 iterative closest point (ICP) algorithm developed concurrently by Besl & McKay [1] and Zhang [20] has been shown to be an excellent method for registration of free form surfaces [16]. 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 iterative closest point verification algorithm avoids local minimums in pose space and handles the outlier problem by boot-strapping an ICP algorithm with previously established point correspondences.

The rest of this paper is organized as follows: Section 2 describes the creation of spin-images from oriented points bases. Section 3 shows how we use spin-images to establish point correspondences between points and verify matches of model to scene. Section 4 discusses the overall robustness of our algorithm and Section 5 shows where we plan to go with the algorithm in the future. It should be noted that this report is the first of three. It discusses the foundations of our algorithm for recognizing objects by matching oriented points. In subsequent reports we will discuss the control of mesh resolution for object recognition using oriented points, and provide a detailed analysis of the use of oriented point matching for object recognition.

2 Spin-Images of Oriented Points

The fundamental shape element we use to perform object recognition is an **oriented point**, a three-dimensional point with an associated direction. Oriented points are similar to oriented particles [18], the main distinction being that the position of oriented points are static, while oriented particles move about the surface of the object based on forces applied by other particles. For surface based object recognition, we define an oriented point O on the surface of an object using surface position $\boldsymbol{p}$ and surface normal $\boldsymbol{n}$. As shown in Figure 1, an oriented point defines a 2-D basis $(\boldsymbol{p},\boldsymbol{n})$ (i.e., local coordinate system) using the tangent plane $\mathcal{P}$ through $\boldsymbol{p}$ oriented perpendicularly to $\boldsymbol{n}$ and the line $\mathcal{L}$ through $\boldsymbol{p}$ parallel to $\boldsymbol{n}$. The two coordinates of the basis are α, the perpendicular distance to the line $\mathcal{L}$, and β the signed perpendicular distance to the plane $\mathcal{P}$. A **spin-map** S_O is the function

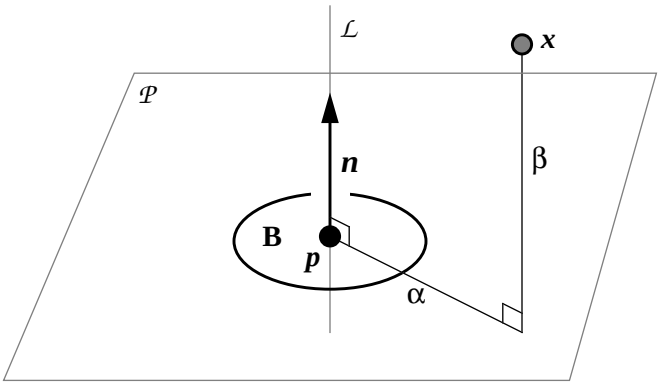

Figure 1: Oriented point basis.

that maps 3-D points $\boldsymbol{x}$ to the 2-D coordinates of a particular basis $(\boldsymbol{p},\boldsymbol{n})$ corresponding to oriented point O

$$S_O: \boldsymbol{R}^3 \rightarrow \boldsymbol{R}^2$$
$$S_O(\boldsymbol{x}) \rightarrow (\alpha, \beta) = \left(\sqrt{\|\boldsymbol{x}-\boldsymbol{p}\|^2 - (\boldsymbol{n} \cdot (\boldsymbol{x}-\boldsymbol{p}))^2}, \boldsymbol{n} \cdot (\boldsymbol{x}-\boldsymbol{p}) \right). \quad (1)$$

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 $\mathcal{P}$ and centered on $\mathcal{L}$ will have the same coordinates (α,β) with respect to the basis.

Except at discontinuities, the surface normal is a well defined axis for a point on the surface of an object because it can be computed without calculating second-order surface derivatives (it is determined as the eigenvector with smallest eigenvalue of the inertia matrix of the surrounding points [4]). If one were to attempt to create an Euclidean three-dimensional basis at a point on the surface of an object using the surface normal as one of the axes, two axes in the plane tangent to the point would have to be determined. An obvious choice for these axes would be the directions of principal curvature on the surface [2], which would result in the construction of the Darboux frame at the point. However, determination of the directions of principal curvature of a surface require the computation of second-order surface derivatives, so the resulting axes are very susceptible to noise. Instead of attempting to calculate a stable 3-D basis at each point, the known stable information (surface normal) is used to compute a 2-D basis. Although a spin-map is not a rigid transformation, 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. Each unique oriented point has a unique spin-map associated with it. In the next section we will describe how we use this fact to encode the shape of objects in an object centered fashion.

2.1 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 other points on the surface of the object $\mathcal{M}$, a set of 2-D points is created. In this case, the points on the surface of $\mathcal{M}$ 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** $\mathbf{I}_{O,\mathcal{M}}$ to refer to the result of applying the spin-map S_O to the set of points on $\mathcal{M}$. 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 to a 2-D space where some of the 3-D metric information is preserved. Figure 2 shows three examples of the result of applying the spin-map at a point on the surface of a femur bone to the other points on a bone to generate three descriptive spin-images.

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. Therefore, spin-images can be used to describe the shape of an object independently of its pose; they are truly object centered shape descriptions.

Suppose we have a method for comparison of spin-images. A simple recognition scheme based on indexing of the spin-images is then possible. Spin-images in the scene are compared to spin-images in the model data base; when a good match occurs, a correspondence between the model point and the scene point 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 sets of 2-D points.

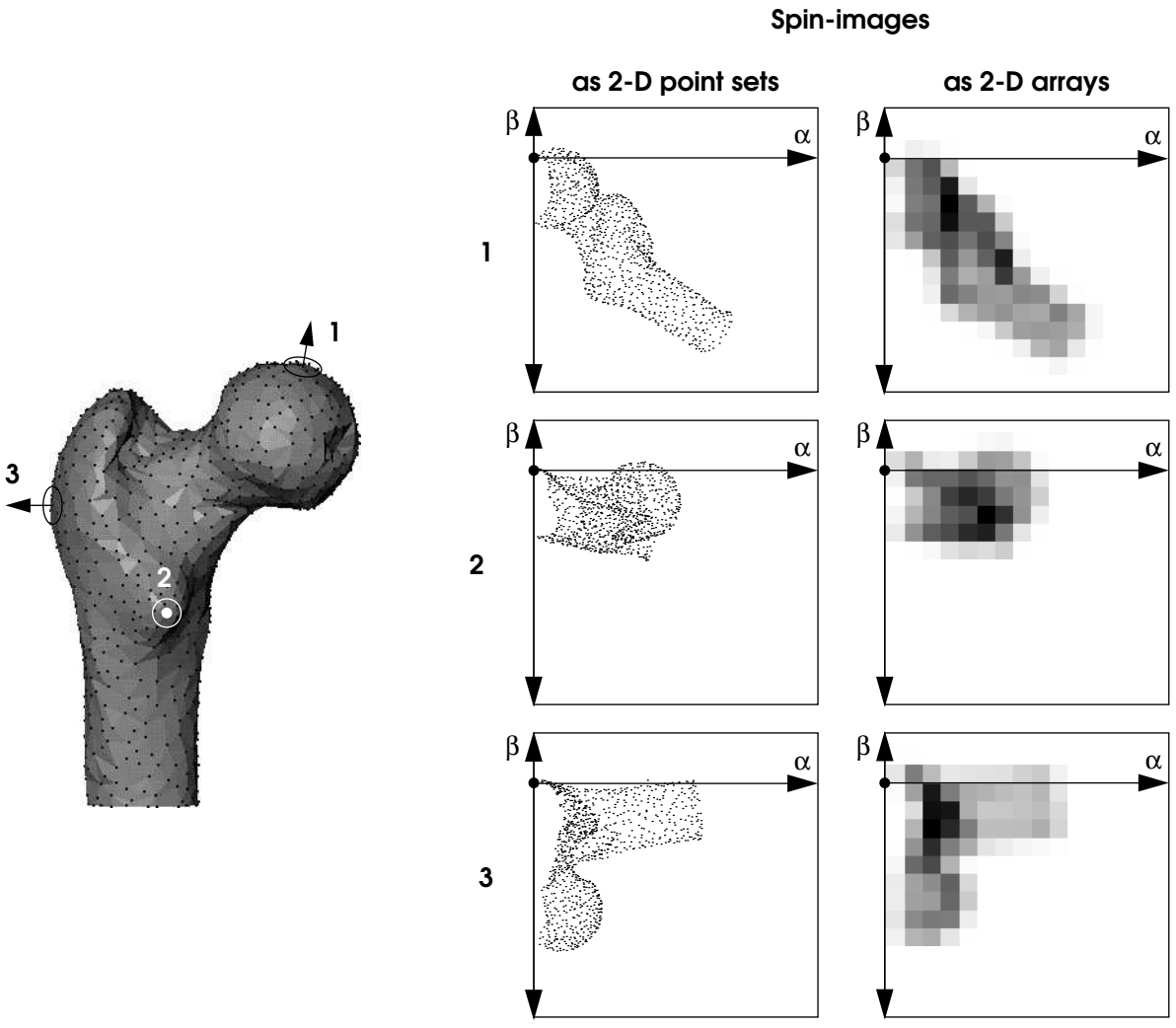

Figure 2: Points distributed on the surface of a femur bone with three oriented point bases and the spin-images associated with the three spin-maps shown as point sets and 2-D arrays viewed as greyscale images.

Originally we approached the problem of object recognition using oriented points with methods from geometric hashing [13][14]. Because we knew it would be too inefficient to compare sets of 2-D points using clustering or nearest neighbors algorithms, our initial algorithm used a hash table to encode geometric constraints between points. Indices to oriented points were stored in the table in the bins determined by spin-map coordinates of other points on the object. Comparison was done by choosing a point in the scene, computing bin locations from spin-map coordinates of 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. Through our studies, we discovered that the number and density of points being stored in the table (because we are using all of the points on the surface of our object and not just a small number of feature points) was defeating the purpose of using a hash table. It seemed that instead of a hash table, a representation more along the lines of those used in structural indexing would be more efficient and appropriate [17].

From our work using a hash table it became apparent that an efficient way of storing and comparing the points of a spin-image is by representing them with a discrete 2-D array instead of as sets of 2-D points. Since the points on the surface of an object will not be the same for different surface mesh representations, we are more concerned with the distribution of points in space that describe the shape of the object than with the position of individual points. By placing the points in discrete bins, we are reducing the effect of the position of individual points while still maintaining the shape of the object.

To create the 2-D array representation of a spin-image the procedure described in pseudo-code in Figure 3 and pictorially in Figure 4 is invoked. First an oriented point O on the surface of an object is selected. Then for each point $\boldsymbol{x}$ on the surface of the object, the spin-map coordinates with respect to O are computed (Equation 1), the bin that the coordinates index is determined (Equation 3), and then the 2-D array is updated by incrementing the surrounding bins in the table. A simple way to

```
CreateSpinImage(oriented_point O, spin_image SI, surface_mesh M)
   for all points x on M
      (α,β) = SpinMapCoordinates(O,x)      // Equation 1
      (i,j) = SpinImageBin(α,β)            // Equation 3
      (a,b) = BilinearWeights(α,β)         // Equation 4
      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 3: Pseudo-code for the generation of a 2-D array representation of a spin-image.

update the table for each point ***x*** would be to increment the bin that the point is spin-mapped to (Equation 4) by one. However, in order to smear 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 processed, a 2-D array representation of the spin-image is generated. Figure 2 shows the 2-D array representations of three spin-images generated for three oriented points on the surface of a femur bone.

We have chosen bilinear interpolation of the spin-map coordinates of the points because it is a simple way to reduce the effects of possible noise in the data. The work of Rigoutsos and Hummel [15] has shown that the statistically optimal method for distributing the contribution of a point in a hash table for 2-D affine point matching is through gaussian weighting of nearby bins based on some expected error in the position of the point. However, because of the nonlinearity of the spin-maps the application of their methods would require computationally expensive searching for bins to increment and a normalization of weights for each point. Therefore, we do not currently use their method, but are considering it for implementing in future versions of our algorithm.

For the rest of this paper we will also use the term spin-image to refer to the 2-D array generated from the spin-mapping of points on a 3-D object. If a differentiation between the 2-D array and the set of points needs to be made, then it will be made clear from context. Before a spin-image can be generated, some important steps that orient the surface of the object and set the size of the spin-images must be taken.

First, consistently oriented surface normals (all pointing out) on the object surface must be determined, so that spin-images constructed from different instances of object are consistent. Since we

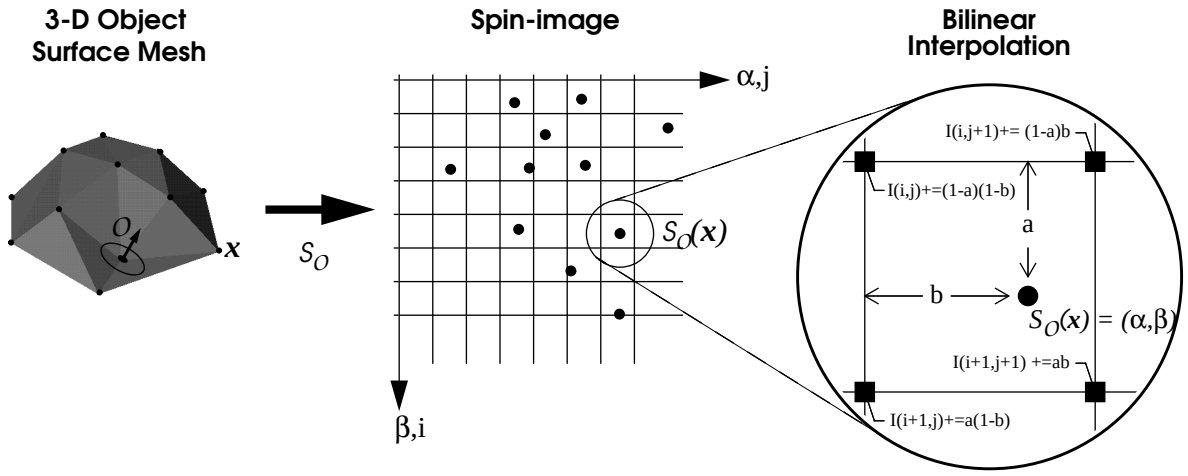

Figure 4: The addition of a point to the 2-D array representation of a spin-image.

are using a surface mesh representation for 3-D objects, we can compute surface normals as the normals of best fit planes (in the least-squares sense) fit to the point and all of its connected neighbors in the surface mesh. We ensure consistent orientation of the normals by picking a surface normal and recursively spreading its orientation to all of the normals of adjacent points. 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 point and the vector from the centroid of the object to the point. If the majority of scalar products are positive, the normals are 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 will not generate outside oriented surface normals although objects can be constructed where it will fail.

In order to ensure that spin-images are large enough to store contributions from all points on the object while not being larger than necessary, the maximum size of the object in oriented point coordinates must be computed. To determine the maximum sizes, oriented point bases are constructed at every oriented point on the object. Then, for each oriented point basis, the spin-map coordinates of all the other points on the object are computed. The maximum α and $|\beta|$ encountered for all of the oriented point bases are the maximum sizes α_{max}, β_{max} of the object in oriented point coordinates.

The size of the bins in the spin-images has to be determined as well. Bin size *b* is very important because it determines the storage size of the spin-image and has an effect on the saliency of the spin-images. The bin size is set as a multiple of the resolution of the surface mesh (measured as the average of the edge lengths in the mesh) in order to reduce the effects of object scale and resolution on this decision. Setting bin size based on mesh resolution is feasible because mesh resolution is related to the size of shape features on an object. The spin-images should blur the position of points while still representing of the structure of the object. We have found that setting the bin size to be two to eight times the mesh resolution yields good results. (Analysis of the effects of bin size on establishment of correspondences is discussed in Section 4 and will also be presented in detail a forthcoming report.)

Once the bin size and maximum size of the object in spin-map coordinates has been calculated, the sizes of the spin-image (i_{max},j_{max}) can be calculated The sizes of the spin-image are

$$i_{max} = \frac{2\beta_{max}}{b} + 1 \qquad j_{max} = \frac{\alpha_{max}}{b} + 1 \quad . \tag{2}$$

Because the distance to the tangent plane of an oriented point can be both positive and negative, the size of the spin-image in the β direction is twice β_{max}. The equations relating spin-map coordinates and spin-image bin (*i*,*j*) are

$$i = \left\lfloor \frac{\beta_{max} - \beta}{b} \right\rfloor \qquad j = \left\lfloor \frac{\alpha}{b} \right\rfloor \tag{3}$$

where $\lfloor f \rfloor$ is the floor operator which rounds f down to the nearest integer. Given the bin size, the bilinear weights used to increment the bins in the spin-image can be calculated

$$a = \alpha - ib \qquad b = \beta - jb\,. \tag{4}$$

Given the size and number of bins in the spin-images and properly oriented surface normals, a spin-image can be generated for any point in the surface mesh representing an object.

2.2 Comparison of 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 the same spin-images. 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 on object with respect to the other. Implicit in this method of recognition is a method for comparison of spin-images.

In general, two different surface meshes representing an object will have roughly the same shape, but will not have the same points on the surface of the object. This can be caused by many factors including different sensor configurations during data acquisition (e.g., different range images of an object) and data acquired using different modalities (e.g., CAD model and a range image). Any point based recognition system that does not extract features cannot make strict assumptions about the position of points on the surface of the object and still be expected to work robustly.

Spin-images do not make strict assumptions about the position of points on the surface of an object. The surface normal at each point is calculated as the best fit plane to a point and all of its nearest neighbors on the surface of the object. This calculation is less sensitive to the position of the points along the best fit plane than it is to the position above or below the plane. In cases where, locally, the surface can be represented as a plane (small curvatures with respect to point spacing), the calculation of surface normal is not sensitive to point position. The spin-images are generated by incrementing bins in a 2-D array based on bilinearly interpolating the position of a point in spin-map coordinates. The bilinear interpolation smears the position of the point in space, making its actual position less relevant to the final appearance of the spin-image. Given many points falling into the same bin in a spin-image, the overall effect of one point will become less important and the consensus of many points will dominate. The requirements for spin-images to be similar for different instances of an object are that surface normals between objects be similar and that the points be

distributed over the surface of the objects in a regular manner. In Section 4 and in a future report, we will discuss the effects of noise, point distribution and bin size on the construction 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

$$R(P,Q) = \frac{N\sum p_i q_i - \sum p_i \sum q_i}{\sqrt{\left(N\sum p_i^2 - \left(\sum p_i\right)^2\right)\left(N\sum q_i^2 - \left(\sum q_i\right)^2\right)}}. \quad (5)$$

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

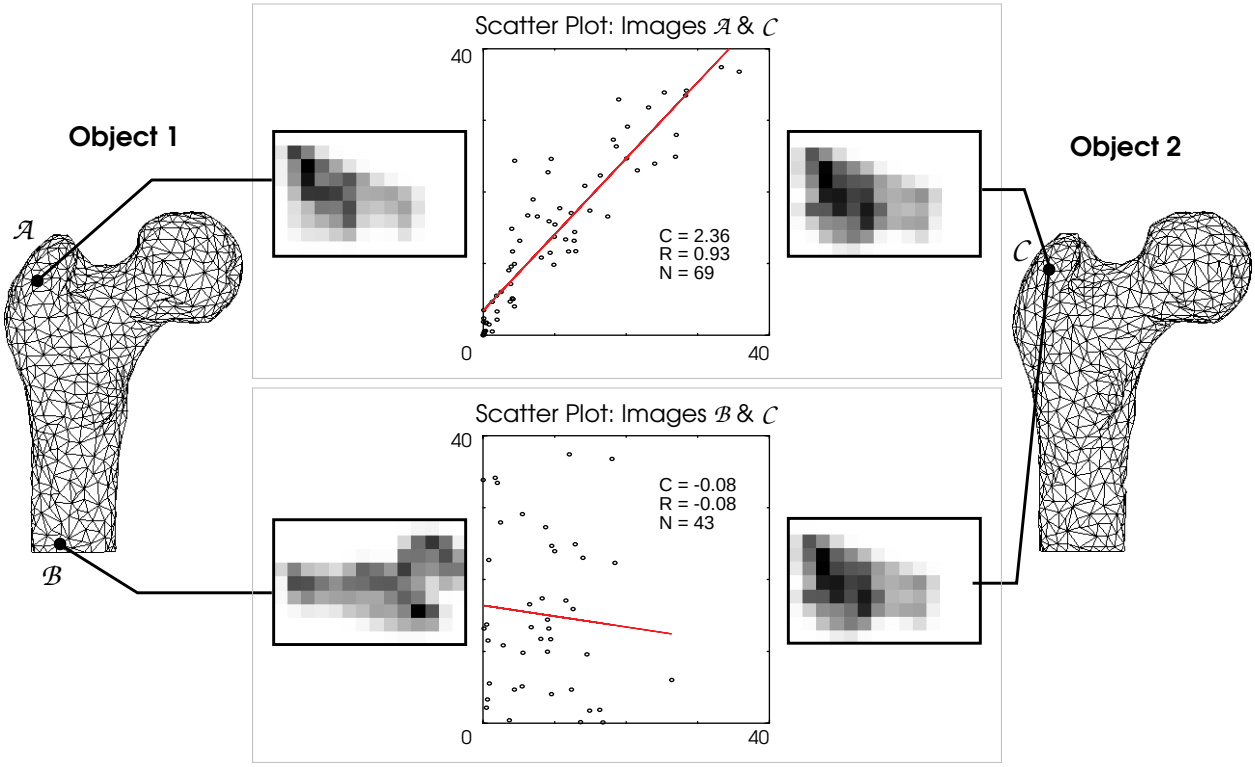

Figure 5: Comparison of spin-images visualized as scatter plots.

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 an 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. 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 method for handling occlusion that involves removing portions of the images from the image comparison. If the a bin in either of the images does not have a value (i.e., no point fell in it), then that bin is not considered in the calculation of the linear correlation coefficient. This technique operates under the assumption that if no value is present in the bin of one image while a value occurs in the other, then there is an occlusion of a surface in the first data set that is not present in the second data set. Knowledge of the image generation process is used to eliminate outliers.

Since the linear correlation coefficient is a function of the number of bins used to compute it, the amount of overlap will have an effect on the correlation coefficients obtained. The more bins used to compute the correlation coefficient, the more confidence there is in its value (i.e., the variance of the correlation coefficient will decrease with more bins). The variance of the correlation coefficient should be included in the calculations of the relative similarity between two images so that the similarity measure between pairs of images with differing amounts of overlap can be compared (for ranking point correspondences). 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. An appropriate similarity function C (derived in Appendix A) which we will use instead of the correlation coefficient to compare spin-images P and Q is

$$C(P, Q) = (\operatorname{atanh}(R(P, Q)))^2 - \lambda\left(\frac{1}{N-3}\right) \quad . \tag{6}$$

This similarity functions weights the correlation coefficient R against the variance in the correlation coefficient, thus including the amount of overlap in the comparison of two spin-images. The hyperbolic arctangent function is used to transform the correlation coefficient into a normal distribution where the equation of the variance is easy to calculate. In Equation 6, λ is a free variable used to weight the variance of the correlation coefficient against the value of the correlation coef-

ficient. In practice we set λ equal to three. If λ is set to zero the similarity function has the same effect as using the correlation coefficient alone, and if λ is very large, the correlation coefficient value become irrelevant and the images are compared based solely on amount of overlap. The similarity function will return a high value for two images that are highly correlated and have a large number of overlapping bins.

Figure 5 illustrates the how spin-images are compared between two different (different points, different connectivity) surface meshes representing a femur bone. On the left, two oriented points $\mathcal{A}$ and $\mathcal{B}$ with associated spin-images are selected from Object 1; on the right, a single oriented point $\mathcal{C}$ with associated spin-image is selected from Object 2. Points $\mathcal{A}$ and $\mathcal{C}$ 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 if two data sets are correlated; the distribution of data points will cluster around a line when the two images are linearly correlated. For points $\mathcal{A}$ and $\mathcal{C}$, the scatter plot of the image data shows high correlation (R = 0.93) and similarity (C=2.36). Points $\mathcal{B}$ and $\mathcal{C}$ are in not in similar positions on the object, so their spin-images are not similar. The scatter plot of the image data shows low correlation and (R = -0.08) and low similarity (C=-0.08).

2.3 Establishing Point Correspondences with Spin-images

With a way of ranking spin-image pairs based on similarity, a method for establishment of correspondences between oriented points in a model database and a scene data set has been developed; we call this algorithm Oriented Point Indexing (OPI). 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 (Equation 6) for each image pair is calculated. The images in the model spin-image stack with high similarity measure when compared to the scene spin-image produce model/scene point correspondences 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 efficient convolution of a portion of the scene with the model. Every point in the model is compared to the scene points using the rapid 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.

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 6. The primary issues that still need to be addressed when establishing point correspondences are: how to randomly select scene points, and how to determine correspondences with large similarity.

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 3-D 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 selected 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 (~20) 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

```
OrientedPointIndexing(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 6: Pseudo-code description of the Oriented Point Indexing algorithm for establishing point correspondences between a model and scene surface mesh using spin-images.

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.

A simple way to select the best model points matching the current scene point would be to select the top percent 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.

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). 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 7 shows a histogram of similarity measures for point $\mathcal{C}$ (from Figure 5) on Object 2 when compared to all of the points on Object 1. Four extreme outliers are detected (points with similarity measure greater

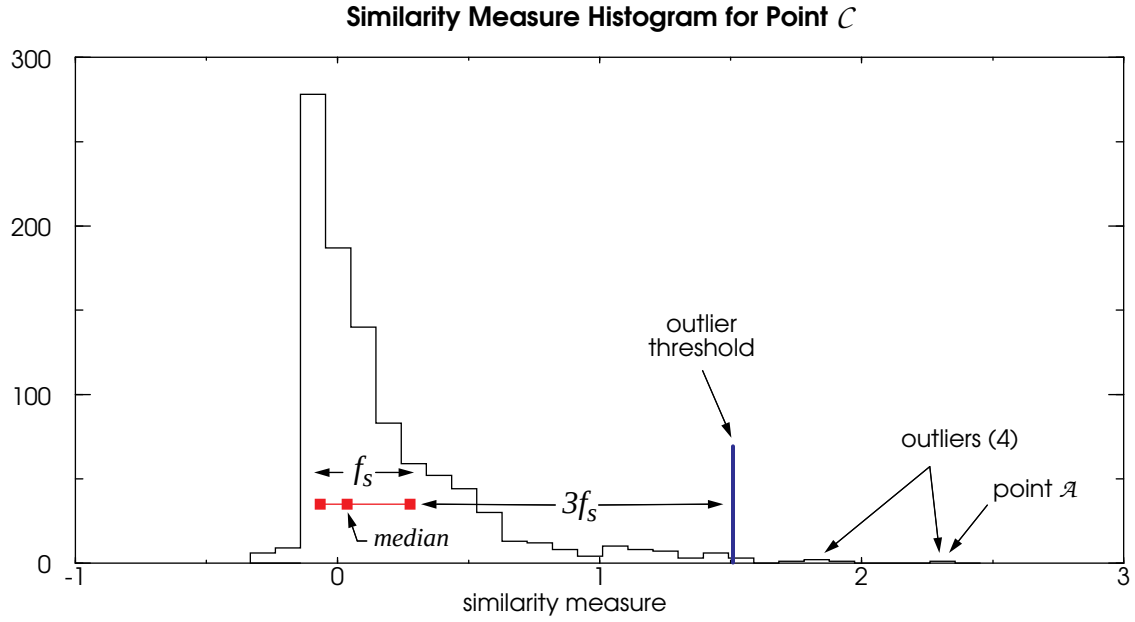

Figure 7: Similarity measure histogram for point $\mathcal{C}$ when compared to object 1 (from Figure 5).

than $3f_s$ above the upper fourth). As expected, the outlier with highest similarity measure is point $\mathcal{A}$ from Figure 5 and the other outliers are points close to $\mathcal{A}$. Through detection of extreme outliers in the histogram of similarity measures, an automatic method for establishing correspondences has been created.

2.4 Handling Partial Views and Scene Clutter

Partial scene views are common in 3-D object recognition because many geometric sensors return a range image than can contain data from only one side of an 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 which will make matching of scene and model spin-images more difficult. By comparing spin-images only in areas of overlap 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.

Suppose there are two oriented points $\mathcal{A}$ and $\mathcal{B}$ with surface normals $\boldsymbol{n}_A$ and $\boldsymbol{n}_B$, respectively, and that $\mathcal{B}$ is being spin-mapped into the spin-image of $\mathcal{A}$. 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

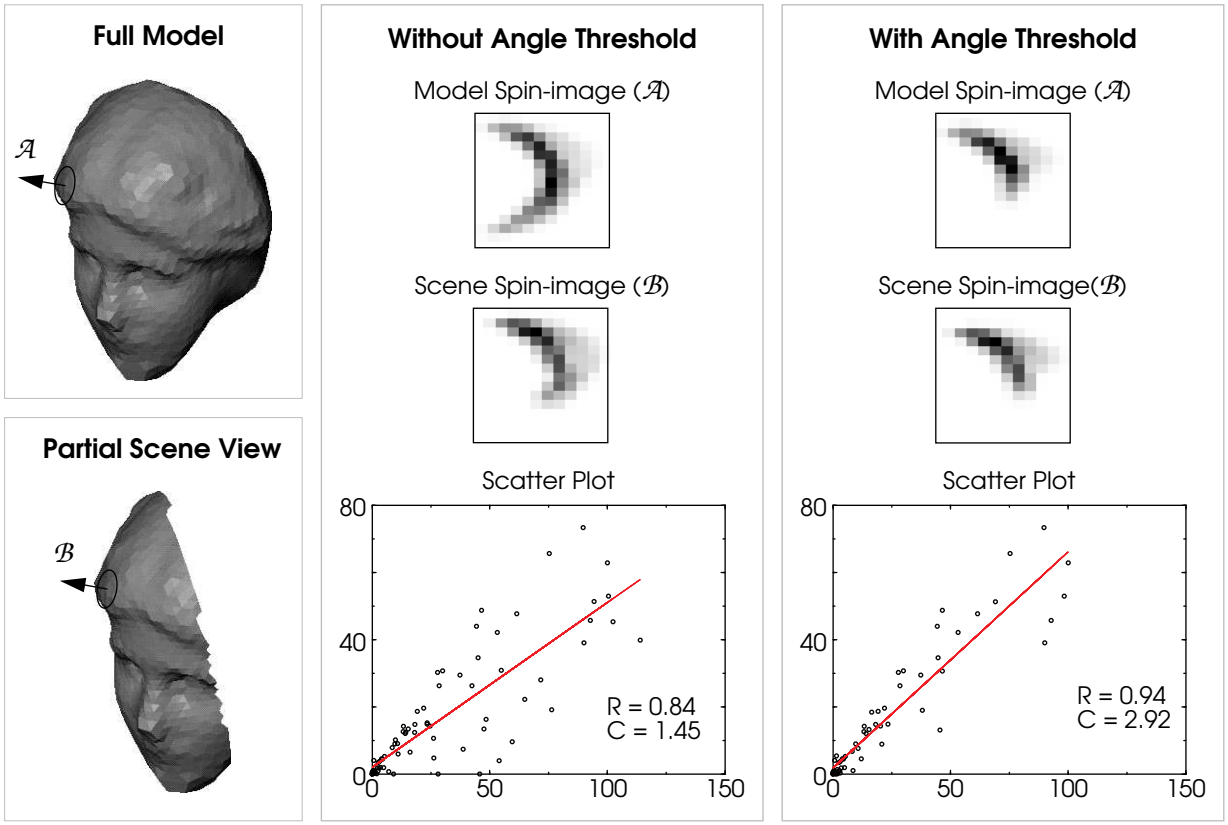

Figure 8: Effect of threshold on angle between surface normals when dealing with partial scene views.

on the viewing direction. For some $\mathcal{B}$ on the surface of the object, the angle between $\boldsymbol{n}_A$ and $\boldsymbol{n}_B$ will be too large for $\mathcal{B}$ to be visible when $\mathcal{A}$ is imaged. Assuming that the viewing direction is oriented along the surface normal of $\mathcal{A}$ (in general this is not true), this angle will be 90 degrees. Using this assumption, all points that have surface normals with angles greater than 90 degrees with respect to $\boldsymbol{n}_A$ should not contribute to the spin-image of $\mathcal{A}$ because they will never be visible when $\mathcal{A}$ is. In the case of partial scene views, applying this angle threshold to the generation of spin-images for the model and the scene will reduce the differences between model and scene spin-images by reducing the areas of overlap in these images where the two might not agree.

In Figure 8, the effect of using this angle threshold in case of the head of venus with a complete model and a partial view scene is demonstrated. When the threshold is not used, the model and scene spin-images for the same point are quite different as demonstrated with greyscale images of the spin-images, their scatter plot and the associated correlations coefficient and similarity measure. However, when an angle threshold of 90 degrees is used, the model and scene spin-images become much more similar as demonstrated with greyscale images of the spin-images, their scatter plot and the associated correlations coefficient and similarity measure.

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 distance that is allowed between the oriented point basis of a spin-image and the points that are contributing to the image. In our implementation, this maximum distance is determined by the size of the model in oriented point coordinates. If a scene point is not spin-mapped into the bounds of the image (as determined by the size of the model spin-images) then the scene point does not contribute to the scene spin-image. Essentially, this distance threshold allows us to place a window on the scene data that is the size of the model from which the scene spin-images are created (comparing spin images is a 2-D convolution of 3-D data at nodes of the scene and model surface meshes). Shorter distance thresholds can be employed to further limit the effects of scene clutter. However, as the distance threshold is decreased, the saliency of the spin-images will decrease as well because less of the overall shape of the object will be encoded. The problem is similar to that of closing window size in correlation based stereo. Future work will investigate the effect of the distance threshold on saliency of spin-images.

A possibly more effective method for limiting the effects of scene clutter on the comparison of spin-images would be to weight the bin values by the inverse of distance (or some other appropriate function) from the oriented point basis when calculating the correlation coefficient of two spin-images. Using this method, points that are far from the oriented point (and consequently are spin-mapped into bins that are far from the origin of the spin-image) would contribute less to the calculation of the correlation coefficient; nearby points would be given more weight than far away points which could correspond to scene clutter. By giving larger weights to points that are more likely to be on the object, scene clutter will have less of an effect on the spin-image comparisons.

3 Recognition from Oriented Point Correspondences

In order to establish a transformation from model to scene, at least 3 point correspondences need to be established. The function `OrientedPointIndexing` drastically reduces the number of possible point correspondences by associating only points in the scene and model that have similar spin-images. However, the function still finds multiple correspondences between scene and model oriented points. There may be more than one model point corresponding to a scene point because of noise in the scene data and symmetries in the model which cause multiple outliers to be detected in the similarity measure histogram. Furthermore, multiple occurrences of the object in the scene could cause more than one scene point to correspond to a single model point. Therefore, it is necessary to filter the correspondences and then group them into sets that are geometrically consistent in order to disambiguate between symmetries, multiple object occurrences and incorrect correspondences. Once this grouping occurs, plausible transformations from model to scene can be calculated and verified to determine the presence of an object in the scene. A complete block diagram of our recognition algorithm described in the following sections is given in Figure 9.

3.1 Filtering Correspondences

If a scene has clutter, not all points in the scene will correspond to the model, or if a scene point has an incorrect surface normal due to noise in the data, it could be matched to model points to which it does not correspond. Therefore, some of the correspondences established by detecting outliers in the similarity measure histograms may be incorrect. In general, these incorrect correspondences will have low similarity measures and overlap when compared to the rest of the correspondences. We employ simple thresholds to remove the incorrect correspondences; if the similarity measure is in the lower percent of similarity measures, or the overlap is in the lower percent of overlaps, then the correspondence is removed. More complicated threshold setting could be used, but it was deemed unnecessary.

3.2 Geometric Constraints on Oriented Points

The first step in grouping oriented point correspondences is to determine if two point correspondences are geometrically consistent. All of the information contained in the oriented points (i.e., position and surface normal) is used to ensure geometric consistency by comparing the spin-map coordinates (Equation 1) of corresponding points. Two oriented point correspondences $[s_1,m_1]$ and $[s_2,m_2]$ are geometrically consistent if their spin-map coordinates are within some distance threshold D_{gc}.

$$\begin{gathered} \left\| S_{m_1}(m_2) - S_{s_1}(s_2) \right\| < D_{gc} \\ and \\ \left\| S_{m_s}(m_1) - S_{s_2}(s_1) \right\| < D_{gc} \end{gathered} \tag{7}$$

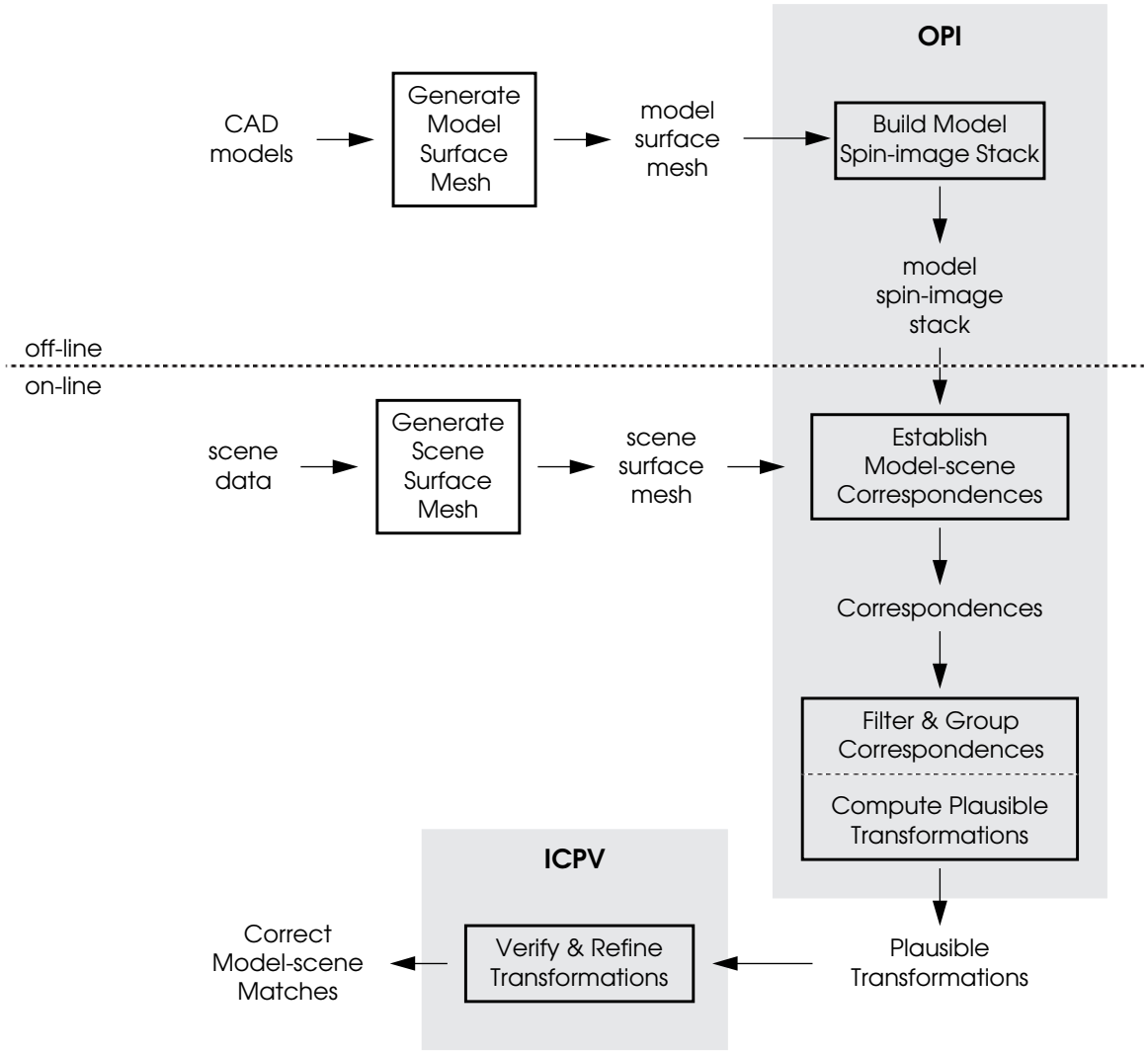

Figure 9: Object recognition block diagram.

This constraint is stronger than simple distance constraints because, by using spin-map coordinates, consistency in distance between points as well as surface normals is checked. Using distance between spin-map coordinates is also an elegant way to combine the constraints on position and normals using a single threshold. In general, we set the threshold D_{gc} to be between one and two times the resolution of the model surface mesh.

3.3 Grouping of Consistent Correspondences

With a method for determining geometric consistency, correspondences can be grouped into plausible transformations. Given a set $C = \{[m_i,s_i]\}$ of point correspondences, the goal is to partition C into groups of geometrically consistent point correspondences. Because we have a ranking of point correspondences, a straightforward and efficient method of grouping can be employed. First the correspondences are ordered by similarity measure. Then the correspondence with the highest similarity measure is selected and all other correspondences that are geometrically consistent with it (Equation 7) from the set C are determined (call this set G). Next, each correspondence in G, in order of similarity measure, is checked against all of the other correspondences in G. If two correspondences are not geometrically consistent then the one with lower similarity measure is removed from G. The end result is a reduced set of correspondences G which are all geometrically consistent with each other and from which a plausible transformation can be calculated. The correspondences in G are removed from C, and the process is repeated with the remaining correspondence in C until no more geometrically consistent groups can be made.

Because the correspondences are selected in order of similarity measure, correspondences with the highest similarity measure are likely to be grouped together. Because the groups of consistent correspondences are removed from the set C, the groups of correspondences will be disjoint. Therefore, this algorithm for grouping correspondences balances the search for all geometrically consistent transformations (a computationally expensive operation) against the desire to find a set of transformations that are more likely than others, given high similarity measures of their associated correspondences.

Given a group $G = \{[m_i,s_i]\}$ of N correspondences between model and scene points, the best transformation T is the one that maps the model points m_i to scene points s_i that minimizes

$$E_T = \sum_i^N \left\| s_i - T(m_i) \right\|^2 = \sum_i^N \left\| s_i - (Rm_i + t) \right\|^2. \tag{8}$$

In the case of a rigid 3-D Euclidean transformation, the best rotation R is found using a closed form solution based on quaternions [5][11]. The best translation t is then

$$t = \left(\sum_i^N s_i \right) - R \left(\sum_i^N m_i \right) \tag{9}$$

A match between scene and model is a set of point correspondences and the associated transformation. Matches are ordered for verification based on the number of correspondences in the match.

3.4 Iterative Closest Point Verification from Correspondences

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. We have developed a method of verification that is a variation of the iterative closest point algorithm. Not only does this method verify possible matches, but it also improves the transformation between scene and model.

The iterative closest point (ICP) algorithm concurrently proposed by Besl & McKay [1] and Zhang [20] 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

---

```
ICPV(match MA, surface_mesh MODEL, surface_mesh SCENE)
While new correspondences are being created
   Apply transformation from MA to points in MODEL
   For each correspondence [s_i,m_i] in the MA
      Access the nearest neighbors to s_i, NN(s_i) in SCENE
      Access the nearest neighbors to m_i, NN(m_i) in MODEL
      For each s_j in NN(s_i) find the closest point m_j in NN(m_i) and make a
         correspondence [s_j,m_j] from it.
      If d(s_j,m_j) < D_v add [s_j,m_j] to MA
   While transformation error is decreasing
      Compute best transformation from correspondences in MA
      Transform model based on best transformation
      For each correspondence [s_i,m_i] in MA
         Access the nearest neighbors to m_i, NN(m_i) in MODEL
         Find the closest point m_j to s_i in NN(m_i)and replace [s_i,m_i] with
            [s_i,m_j] in the current match if d(s_i,m_j) < d(s_i,m_i) < D_v
```

Figure 10: Pseudo-code description of the Iterative Closest Point Verification algorithm.

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 point correspondences using the OPI algorithm.

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. The iterative closest point verification (ICPV) algorithm does this by growing closest point correspondences from initial correspondences given by the OPI algorithm.

Verification starts with an initial list of point correspondences from which the transformation of model to scene is computed and then applied to the model points. Next, for each correspondence, new correspondences are established between the nearest neighbors of the model point and nearest neighbors of the corresponding scene point if the distance between closest points is less than a threshold. This step grows the correspondences from those correspondences already established by finding scene points that correspond to model points. The transformation based on the new correspondences is computed and then refined using traditional ICP by moving the model by the new transformation and recalculating closest points and recomputing the new transformation until the transformation error stops decreasing. The growing process is repeated until no more correspondences can be established. The final transformation used is the composition of all of the transformations computed in the verification algorithm. The ICPV algorithm grows patches of correspondence between the model and the scene from the initial correspondences. A cascade ef-

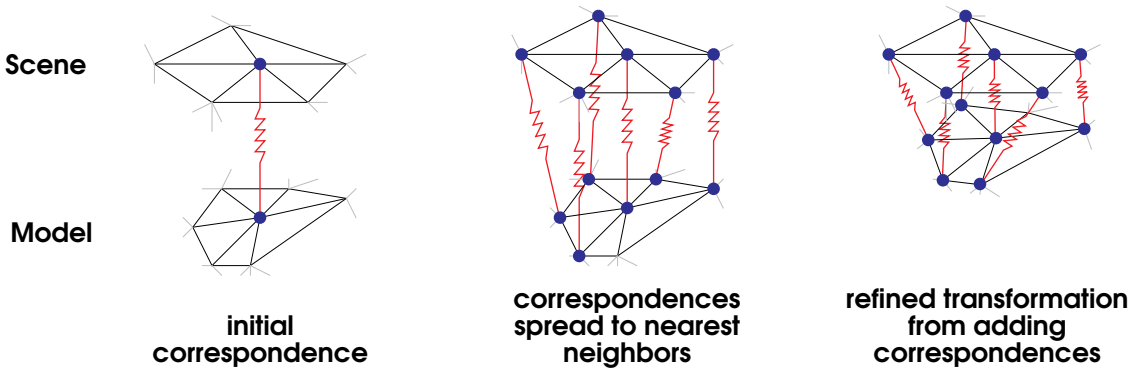

Figure 11: Illustration of the spreading of point correspondences and refinement of the match transformation in the ICPV algorithm.

fect occurs. If the match is good, a large number of points will be brought into correspondence; if the match is bad, the number of correspondences will remain close to the number established by the OPI algorithm. Therefore, a good measure of the validity of the match is the number of correspondences after verification. Figure 11 illustrates how a single point correspondence is spread to nearest neighbors in the surface mesh and how the current match transformation is refined. A complete pseudo-code description of the ICPV algorithm is given in Figure 10.

Although the OPI and ICPV algorithms can be formulated using various spatial data structures (k-d trees, surface meshes, volumetric oct-trees), in practice all of the scene and model data are represented as surface meshes. The approximate nearest neighbors to a point can be efficiently determined from a surface mesh by using the points that are connected to the point in question by the surface mesh. The validity of using adjacent points in the surface mesh as nearest neighbors is dependent on the mesh generation process. A possible drawback of using surface meshes in the ICPV algorithm is that correspondences will not be established between surface mesh regions that are not connected to regions that contain correspondences.

The threshold D_v in the verification stage (that sets the maximum distance by which two points can differ and still be brought into correspondence) can be set automatically if the resolution and noise in the data is known. In the case of no noise and equal spacing, D_v should be set to the resolution of the mesh. Since the points are not generally distributed equally and noise exists in the data, D_v is set to between one and three times the resolution of the mesh. This is a compromise between allowing for noise and preventing correspondences in regions where the data sets do not overlap. Figure 12 illustrates how initial correspondences established by the OPI algorithm are spread over the surfaces of two range views (taken with a structured light range camera) of a plastic model of

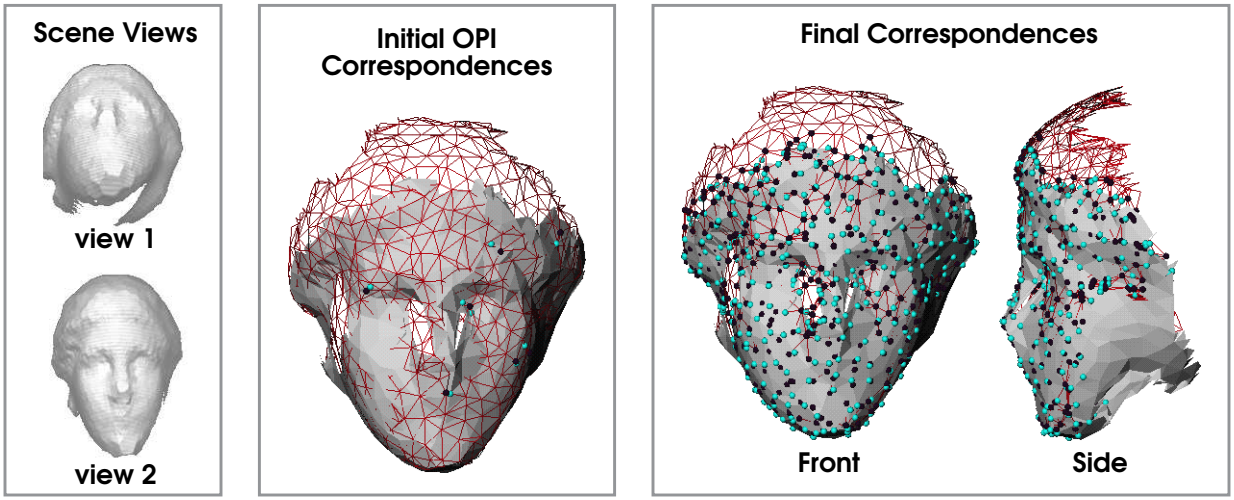

Figure 12: The registration of two views of a plastic model of the head of Venus. From left to right are shown: two camera images of the views to be registered, the registration of the views (view 1 shaded, view2 wireframe) from OPI correspondences shown as spheres, and a frontal and side view of the registration after the ICPV algorithm with established correspondences.

the head of the goddess Venus. The correspondences are established only in the regions where the two surface meshes overlap, thus preventing a poor registration caused by correspondences being established between non-overlapping regions.

3.5 Simultaneous Multiple Model Recognition

In order to be versatile, object recognition systems must be able to recognize multiple objects. Generally this is accomplished by storing representations of many objects in a model database which is then searched for the appropriate model. For multiple object recognition to be useful, the recognition system must be able to discriminate between different objects automatically. This includes the ability to determine when an object appears in the scene, when an object does not appear in the scene and when no objects stored in the model database appear in the scene. The ability to discriminate between different objects is a good measure of how useful a model representation is for object recognition. Spin-images have significant discrimination power, so they can be readily used for multi-model object recognition.

The procedure for multiple model object recognition using spin-images is very similar to recognition of a single object. First a set of points is selected from the scene. Then, for each scene point, the scene spin-image is correlated with all of the model spin-images for all of the models in the database. The similarity measure histogram for the scene point is computed and outliers are detected as in Section 2.3. In general the outliers in the similarity measure histogram will be points on the model that is associated with the scene point and will not be points that come from some other model. Once the outliers for every scene point have been computed, the correspondences are filtered as in Section 3.1. The correspondences are then partitioned based on model; geometrically consistent correspondences and verified transformations are then computed from each partition. Because correspondences are chosen as the outliers in the similarity measure histogram for all models, the correct model is easily selected from the model database. Of course ambiguity will occur if two models are very similar in shape. (We are currently investigating ways to measure shape similarity between models represented by spin-images and our results will be included in a future report.)

3.6 Results

The combination of Oriented Point Indexing and Iterative Closest Point Verification results in an object recognition paradigm that can be used to solve many problems in recognition and registration. To fully demonstrate the effectiveness of our algorithm, we have produced results in two domains: anatomical recognition and interior modeling from range images. Results from anatomical recognition demonstrate that our algorithm can handle complex and free-form objects, while the results from interior modeling demonstrate that the algorithm can also handle more simple parametric objects and register multiple range views.

An initial test of our 3-D object recognition system is to recognize an object represented as a 2 1/2-D surface mesh, generated from a range image of the object, in a 2 1/2-D cluttered scene. Figure 13 shows the result of recognizing a rubber duck in a cluttered scene containing the duck, a shell, a kazoo, a funnel and a model of the head of Venus. First a model surface mesh of the duck is created from a range image (generated by a structured light range camera) of the duck by connecting pixels in adjacent rows, columns and diagonally across rows and columns. This surface mesh is smoothed using the low-pass mesh smoothing filter of Taubin [19] and then decimated in a way that controls the length of edges in the mesh [12] in order to reduce the amount of points that have to be processed while still preserving the shape of the object. The same procedure is applied to the scene range image; then the duck model is recognized in the scene using the OPI and ICPV algorithms. As with all of our recognition results, the number of correspondences established between the model and scene after the ICPV algorithm is a large fraction of the total number of points in the model, so the model is considered to be correctly recognized. How large this fraction has to be depends on the amount of the model that is present in the scene, a quantity that cannot be directly measured. (Future investigations will look into determining this fraction automatically.) The views of the resulting registration of the duck show that it has been accurately positioned in the scene even in the presence of scene clutter and occlusion.

A more difficult recognition task is to recognize a a complete 3-D model in a 2 1/2-D data set with scene clutter. Figure 14 shows the result of recognizing a pipe T-joint model in a range image. The T-joint model was created using CAD techniques and the range image was generated using a Perceptron scanning laser range finder. This example of recognition demonstrates how models can be recognized and localized automatically in an industrial facility to aid in the three dimensional map-

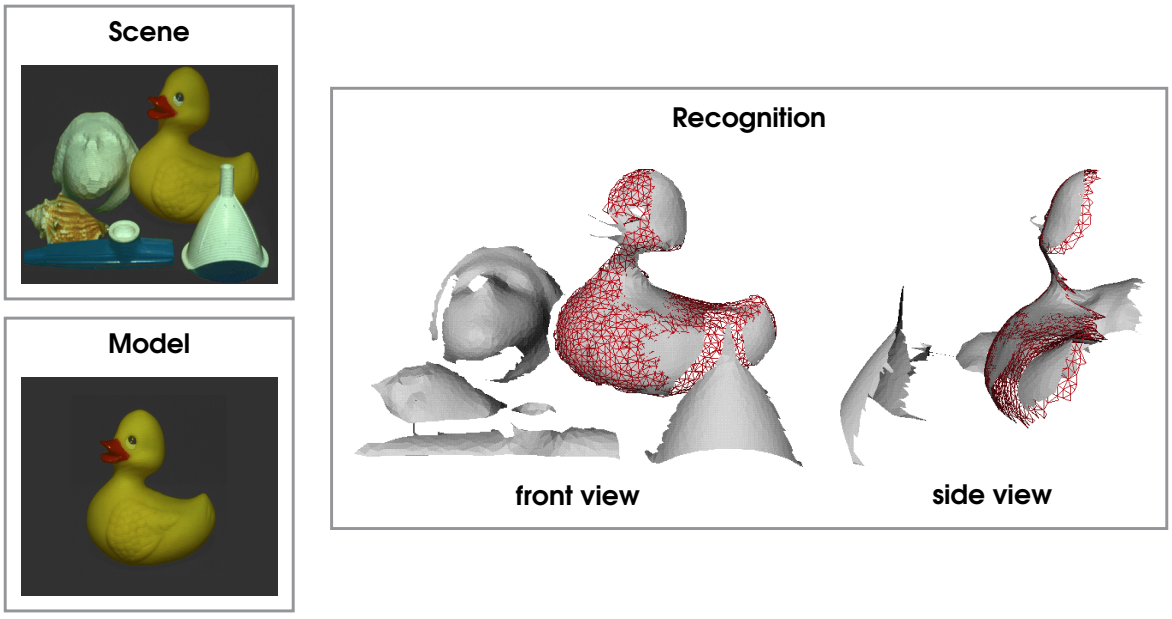

Figure 13: Recognition of a partial duck model in a partial view with clutter and occlusion.

ping of its interior. Figure 14 also demonstrates that our recognition algorithm works even when the resolution of the model and scene are different.

The T-joint that is recognized in Figure 14 is symmetric about a plane passing through the axes of both cylinders making the joint. Symmetries have the effect of increasing the number of point correspondences that have to be processed for recognition because symmetric points on the model will match the same scene point. Fortunately, during the grouping of geometrically consistent correspondences, the disjoint sets of correspondences that arise from symmetry will be separated. Symmetry increase the amount of computation needed for recognition, but it does not prevent recognition as is shown by the result in Figure 14. (In a future report, we will give a detailed analysis the effect of symmetry on oriented point matching.)

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 algorithm provides a technique for determining the transformation between range views when it is unknown or highly uncertain. Figure 15 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 laser range finder at each position. The position of the cart varied each time by approximately 30 degrees of rotation about the vertical axis. Figure 15 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 com-

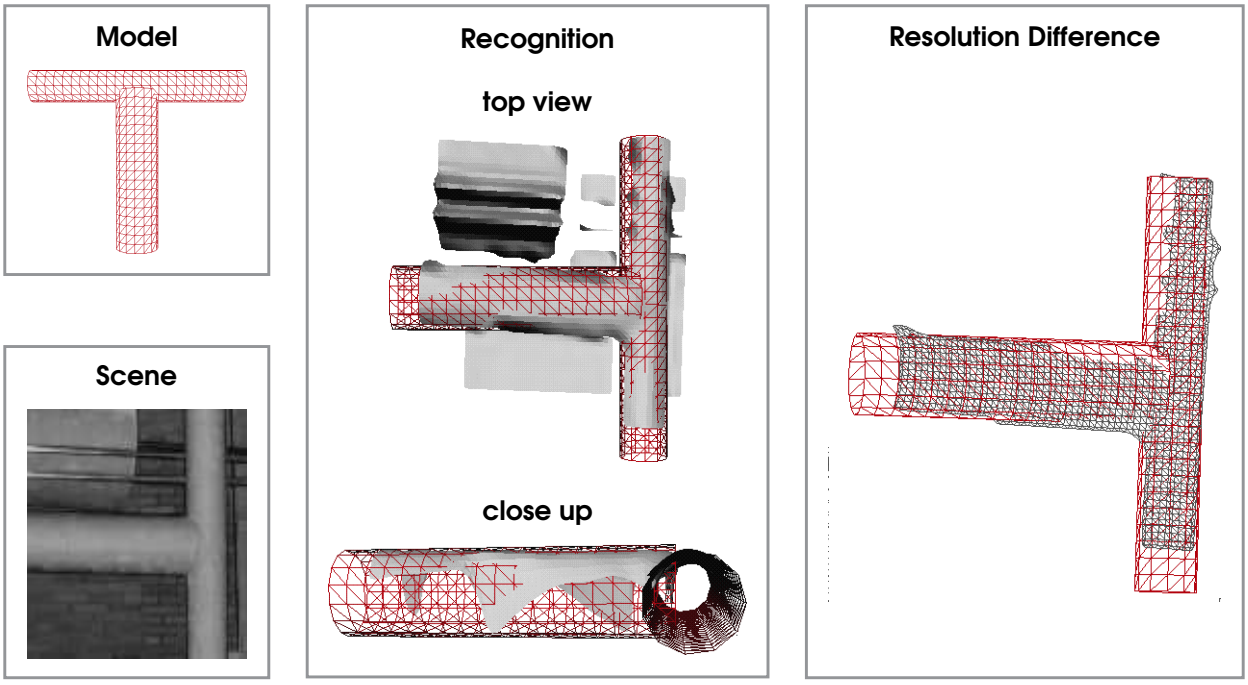

Figure 14: Recognition of a complete pipe T-joint model in a partial view with clutter.

plete scene description than is available from one range image is generated and that the registration is correct up to the resolution of the points.

An important test to demonstrate that our algorithm can be used for efficient object recognition is the simultaneous recognition of more than one complete model in a partial view of a scene. Figure 16 shows a recognition result where complete femur and pelvis bone models are simultaneously recognized and located in a range image of a scene containing a femur and a pelvis. The complete models of the femur and pelvis were created from CT scans of the objects followed by contour extraction, contour to surface mesh conversion and surface mesh decimation. The range image was acquired using a structured light range camera. Figure 16 shows that the femur and pelvis are properly recognized and located even when the femur occludes the pelvis and the significant portions of the model are missing.

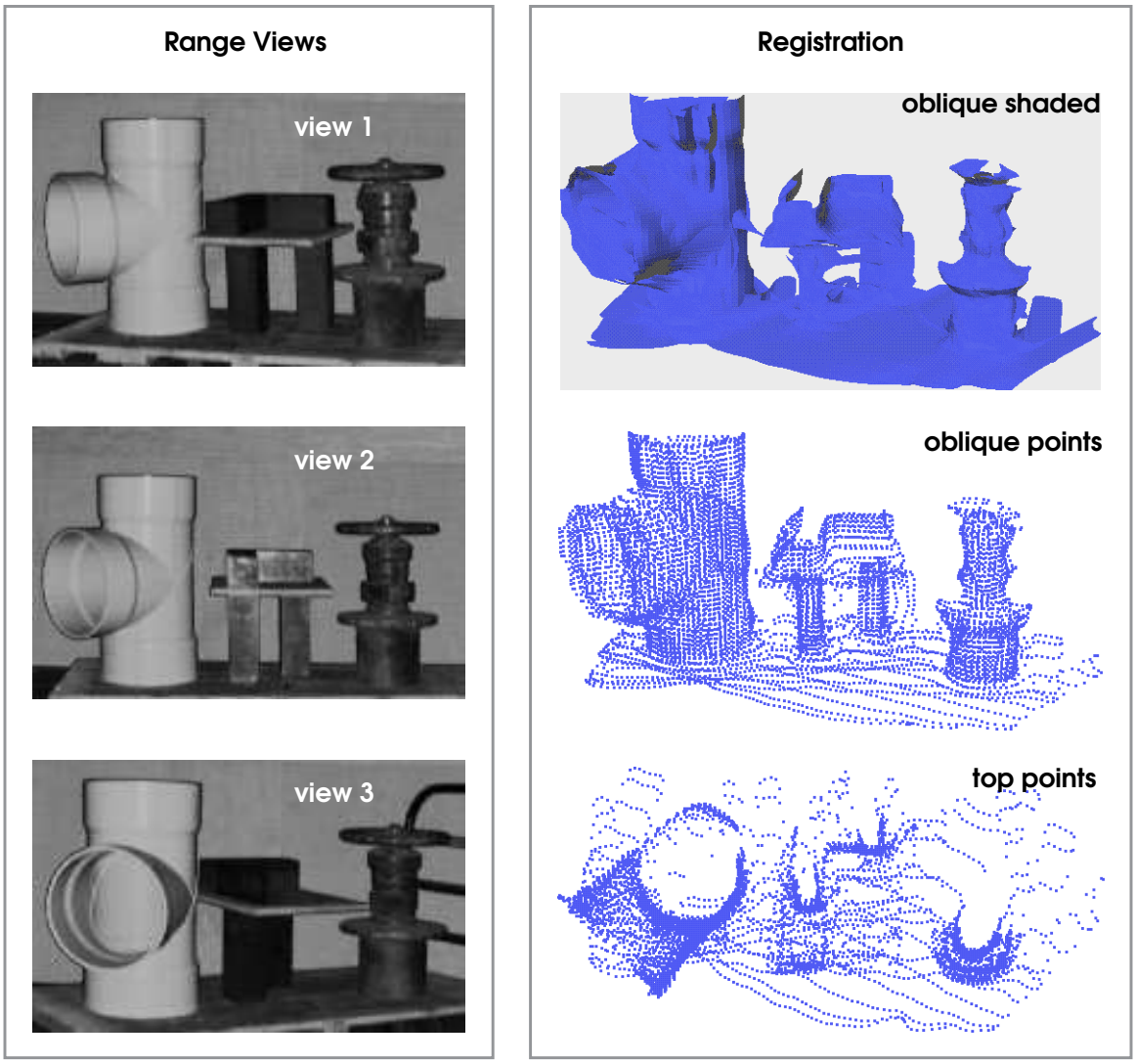

Figure 15: Registration of three range images.

A popular result in the biomedical imaging literature is the registration of two skull data sets generated from volumetric medical scanners [8]. Figure 17 shows the registration of two different surface meshes of a skull created from the same CT data set. The surface meshes were created by adding small amounts of Gaussian noise to the points in the original surface mesh generated from the CT scans and then decimating. Since different random noise values were added to the original surface mesh for each of the decimated surface meshes shown, the decimation will create surface meshes with different points and connectivity. A close look at the two wireframe data sets in Figure 17 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 17 the translation is $[-0.053\ 1.981\ -1.768]^T$ mm and the fixed rotation angles are [0.031 -1.092 -1.256] degrees. This corresponds to a translation error

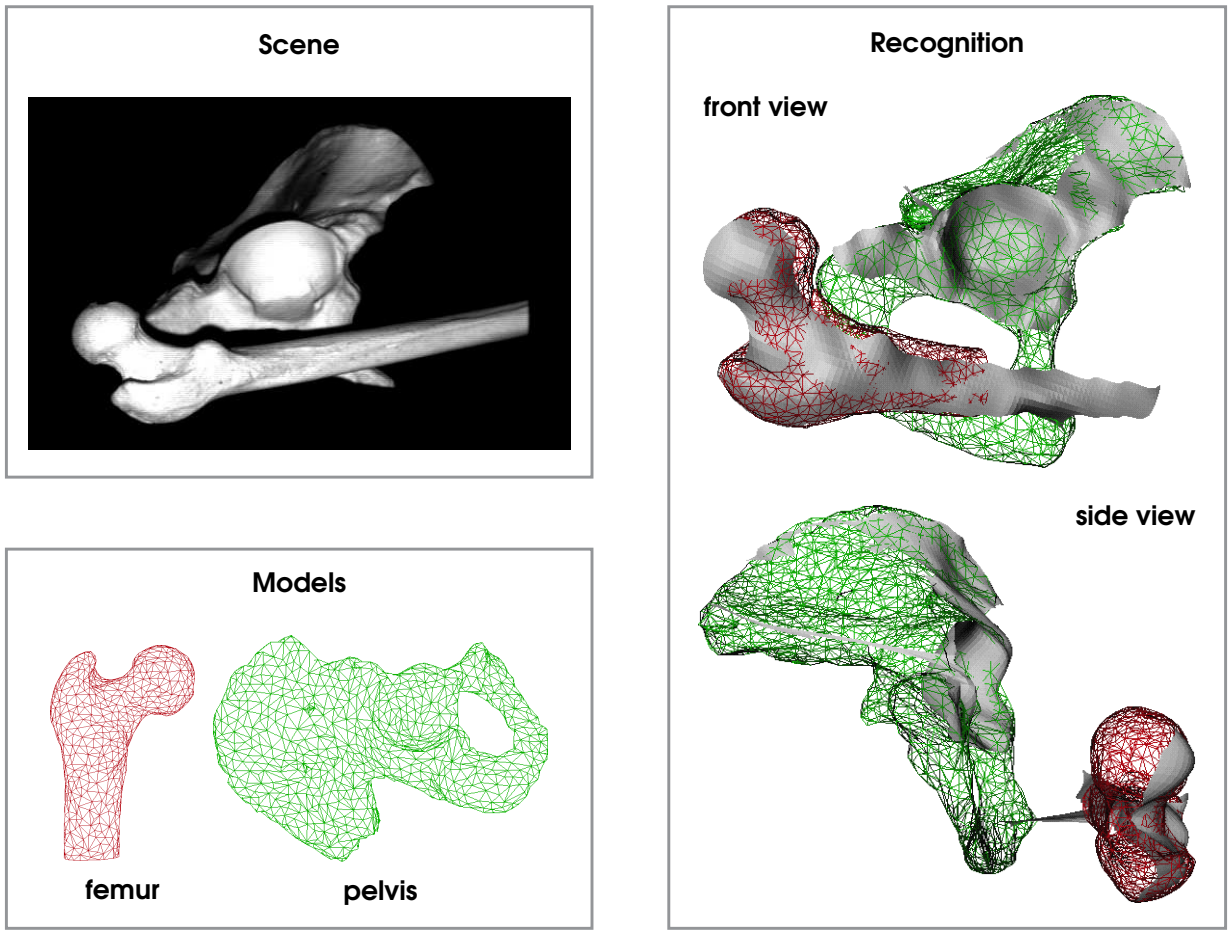

Figure 16: Simultaneous recognition of multiple anatomical models in a partial view with clutter and occlusions.

magnitude of 2.656 mm, which is less than half the resolution of the surface meshes (6.4 mm) used in the registration, and an angular rotation magnitude of 1.665 degrees.

With our recognition algorithm, the registration results cannot be better than the resolution of the surface meshes because the transformation is based on discrete point matching. To improve the registration results beyond the resolution of the points in the data sets, we could follow our verification algorithm by an ICP registration algorithm that includes facet interpolation in the closest point computation. For any system that requires very accurate localization (e.g., medical registration, object recognition for manipulation) this step will be necessary. Furthermore, all of the results shown in this paper were based on matching a model to subsets of the scene data (usually obtained by sub-sampling a range image) in order to decrease computation times. For these results, more accurate registration would be obtained by using all of the data in a final facet based ICP algorithm.

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 Table 1, the first free variable (bin size) is the size of the bins used to create spin images for the model expressed in units of model resolution (the average length of edges in the surface mesh). The next free variable (T_{angle}) is the threshold on difference in angle between surface normals used when creating spin images. N_{sp} is the number of scene points selected randomly from the scene. T_{sim} is the percentage of possible correspondences eliminated based on similarity measure and $T_{overlap}$ is the percentage of possible correspondences eliminated based on overlap when filtering the possible correspondences before making geometrically consistent matches. D_{gc} is the geometric consistency threshold and T_{ICPV} is the threshold on search distance in the ICPV algorithm where both are expressed in units of mesh resolution.

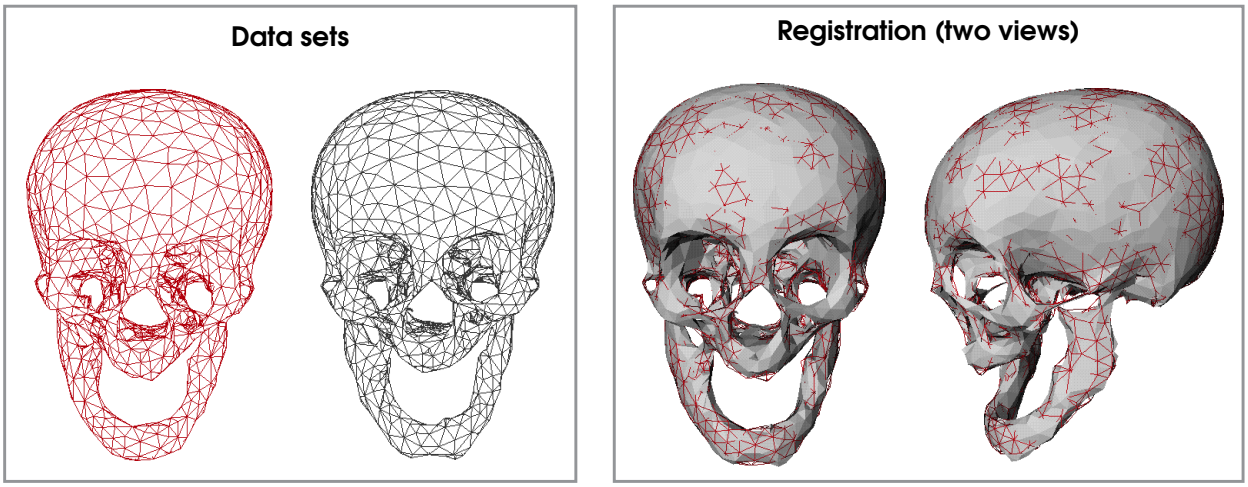

Figure 17: Anatomical registration of a skull.

Table 1: Thresholds and free variables for recognition.

result	bin size	T_{angle}	N_{sp}	T_{sim}	$T_{overlap}$	D_{gc}	T_{ICPV}
duck	3	180	400	50%	50%	2	2
T-joint	3	120	200	50%	50%	1.5	1.5
Range Images	3	180	200	50%	50%	2	2
Pelvis & Femur	3	90	400	25%	25%	2	2
Skull	3	180	200	50%	50%	2	2

4 Discussion

Recognizing objects 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. 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.

4.1 Spin-images

The purpose of a spin-image is to describe a point so that it can be compared to other points. This description is based on the shape of the object encoded by other points on the surface of the object. Because the spin-image is created with respect to an oriented point basis, it is an object centered description that depends on the shape of the object and the position of the point on the surface of the object. Since spin-images depend only on object shape and are created with respect to a coordinate system on the object, they are pose independent. Therefore, they can be used to establish correspondences between points on objects in two different positions.

For the 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 neigh-

borhood 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 only used 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.

It is not necessary for the model and the scene to have the same resolution because of the normalization in the correlation coefficient that is used in the similarity measure when comparing spin-images. However, the distribution of points on the surfaces of the model and the scene need to be proportional. In other words, if the model and scene have varying distributions of points over their surfaces, the variation over one surface needs to be linearly proportional to the variations over the other. If this is the case, then the local resolutions in the model and the scene will always be proportional. In practice it is desirable to have the same resolution over the entire object (a constant distribution of points on the surface) so that accurate estimates of registration error can be made.

As noise is added to the scene, the position of the scene points will change, thus changing the oriented point bases in the scene. Figure 18 shows the spin-images for a fixed basis at increasing levels of noise along with the corresponding objects. As noise is added, the appearance of the spin-images change linearly with the magnitude of the noise; however, they still remain correlated with the corresponding model spin-image even at high levels of noise (on order of the mesh resolution) because the contribution of the each point is spread over bins. Excessively noisy points (outliers) will change only a few pixels in the image, so the spin-images will remain similar. Establishment of the correspondences is more sensitive to the differences in surface normal between scene and model than it is to the position of the points. However, surface normals are generally calculated

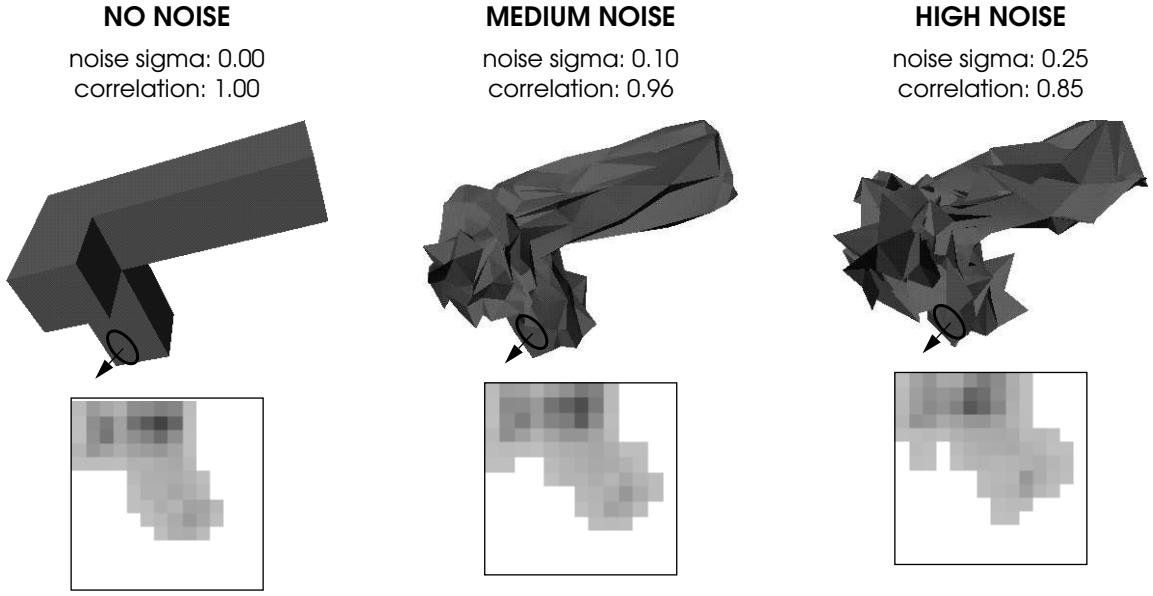

Figure 18: How scene noise effects spin images.

using an average of points on the surface, so they will not be as susceptible to the noise in individual points.

4.2 Robustness of Correspondence Grouping

The similarity measure calculated for each point correspondence can be used to rank correspondences based on their correctness. In the OPI algorithm, the similarity measure is first used to rank correspondences between a single scene point and all of the model points in order to chose the candidate model points corresponding to the scene point. Once the possible corresponding model points have been determined for all of the selected scene points, the similarity measure is used to rank these correspondences. This second ranking is used to eliminate possible point correspondences where the scene data is not close to the shape of the model. Situations where the scene is not like the model occur when: a point that is not on the object is chosen from the scene, part of the scene is excessively noisy, a point on the object is too close to clutter in the scene, or a partial view or occlusion removes data from the scene. By ranking and removing the correspondences of low similarity measure, many correspondences with scene points that do not match the model can be eliminated from the list of possible correspondences. This reduces the total number of correspondences and hence the total number of transformations that have to be verified, thus reducing the running time of the algorithm.

4.3 Computational Complexity

Because oriented point indexing 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 selected from the scene, M the number of model points and I the size of the spin-images. The time to generate the model hash table (which can be done off-line) is $O(M^2)$ 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 table is O(MI) because there is one spin-image for every model point. The establishment of correspondences between the model stack and the scene is O(SMI+SMlogM) because each scene point spin-image must be pixelwise multiplied with all of the model spin-images (O(SMI)) and the M similarity measures of the correspondences must be sorted (O(SMlogM)). Since logM is usually much less than I, the establishment of correspondences can be reduced to O(SMI). The iterative closest point verification algorithm is worst case O(M) for each iteration of the algorithm. This assumes that all of the model points are brought into correspondence with scene points. The computational complexity of the algorithm is not prohibitive as is born out in the recognition times shown in Table 2 for the results shown in Figure 13 through Figure 17.

Table 2: Recognition timing statistics.

model result	# model points (M)	total # scene points	# scene points selected (S)	spin image size (I)	model stack size in bytes	OPI time	ICPV time
Duck	1206	2339	400	264	1.3 M	53 s	7 s
T-joint	571	3306	200	98	0.2 M	19 s	83 s
Range Images	2708	2557	200	264	2.9 M	50 s	130 s
Pelvis & Femur	2331	2807	400	200	1.9 M	84 s	78 s
Skull	2603	2620	200	200	2.1 M	43 s	54 s

5 Future Work

We have presented a new representation for 3-D object recognition that is based on indexing 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 industrial and medical settings where complete models have been recognized in partial scenes from a library of object models.

However, many issues relating to spin-images need to be investigated. Of particular importance is the role of resolution on spin-image generation and comparison. We believe that it will be possible to construct a coarse-to-fine recognition strategy through the control of surface mesh resolution and spin-image size that will provide faster and more accurate registration. Another avenue of investigation will be the organization of model spin-images to provide for more rapid recognition. If the spin-images can be arranged in a tree structure so that only a small number of spin-images from the model need to be compared to find correspondences to each scene spin-image, then more rapid recognition will result. The concepts used to generate spin-images can possibly be applied to other domains in computer vision, namely 2-D scale invariant curve matching and 3-D intensity image registration. We plan to investigate recognition in these domains as well. Spin-images characterize the shape of objects and can be used to compare two objects to see if they are similar. We also plan to investigate methods for comparing the shape of objects using spin-images.

Appendix A: Derivation of 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 scatter plot 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 two 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 bins. To incorporate this confidence when comparing spin-images, the variance of the correlation coefficient must be added to the spin-image similarity measure.

When two spin-images are compared, the calculated correlation coefficient R is drawn from a statistical distribution of correlation coefficients with mean $\overline{R}$ estimated by R. Unfortunately, the distribution of correlation coefficients is not normal. However, if R is transformed into a different variable V

$$V = \text{atanh}\,(R)$$

a normal distribution $N(V,\sigma_V)$ with mean and variance

$$\overline{V} = \text{atanh}\,(\overline{R}) = \text{atanh}\,(R) \qquad \sigma_V^2 = \frac{1}{N-3}$$

results [3].

If all of the comparisons of spin-images had the same number of samples, an appropriate way to pick the best comparison would be to chose the comparison i that satisfies

$$\max_i \left(E\left[V_i^{\,2} \right] \right)$$

where $E[x]$ is the expectation of x. To include the variance (hence, the confidence in the measurement), a feasible similarity measure is one that weighs correlation against the number of samples used to calculated correlation such as

$$\max_i \left(E\left[V_i^{\,2} - \alpha\sigma_{V_i}^2 \right] \right) = \max_i \left(E\left[V_i^{\,2} \right] - \alpha\sigma_{V_i}^2 \right).$$

Since

$$\sigma_V^2 = E\left[V^2 \right] - E\left[V \right]^2$$

this reduces to choosing

$$\max_i \left(E\left[V_i \right]^2 - (1-\alpha)\,\sigma_{V_i}^2 \right) = \max_i \left(\overline{V}_i^2 - (1-\alpha)\,\sigma_{V_i}^2 \right).$$

Converting back to correlation coefficient and substituting $\lambda=(1-\alpha)$, the same relation as in Equation 6 for the similarity measure between two spin-images is obtained.

$$C = \left(\text{atanh}\,(R)^{\,2} - \lambda \frac{1}{N-3} \right).$$

Acknowledgments

This work was performed as part of the Artisan Project in the Field Robotics Center and would not have been possible without the vision of Jim Osborn and the constant technical support from Regis Hoffman. We would like to thank David Simon, Fabio Cozman and Paul Heckbert for many enlightening conversations on registration, statistics and low-level representations. We would also like to thank André Guéziec for providing us with the skull data set.

References

[1] P. Besl and N. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, 1992.

[2] C. Chua and R. Jarvis, "3-D free-form surface registration and object recognition," *Int'l J. Computer Vision*, vol. 17, pp77-99, 1996.

[3] J. Devore, *Probability and Statistics for Engineering and Sciences*, Brooks/Cole, Belmont, CA, 1987.

[4] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley-Interscience, New York, 1973.

[5] O. Faugeras and M. Hebert, "The representation, recognition and locating of 3-D objects," *Int'l J. Robotics Research*, vol. 5, no. 3, pp. 27-52, 1986.

[6] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, Cambridge, MA, 1993.

[7] W.E.L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints*, MIT Press, Cambridge, MA, 1990.

[8] A. Guéziec and N. Ayache, "Smoothing and matching of 3-D space curves," *Int'l J. Computer Vision*, vol. 12, no. 1, pp. 79-104, 1994.

[9] M. Hebert, J. Ponce, T. Boult and A. Gross, Eds., *Object Representation in Computer VIsion*, Springer-Verlag, Berlin, 1995.

[10] 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.

[11] B.K.P. Horn, "Closed-form solution of absolute orientation using unit quaternions," *J. Optical Soc. Amer.*, vol. 4, no. 4, pp. 629-642, 1987.

[12] A. E. Johnson and M. Hebert, "Control of mesh resolution for 3-D object recognition," Carnegie Mellon Robotics Institute Tech. Report CMU-RI-TR-96-20, October 1996.

[13] 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.

[14] 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.

[15] 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.

[16] 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)*, pp. 2235-2241, 1994.

[17] 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.

[18] R. Szeliski, D. Tonnensen and Dimitri Terzopoulos, “Modeling surfaces of arbitrary topology with dynamic particles,” *Proc. Computer Vision and Pattern Recognition 1993 (CVPR ‘93)*, pp. 82-87.

[19] G. Taubin, “A Signal processing approach to fair surface design,” *Proc. Computer Graphics 1995 (SIGGRAPH ‘95)*, pp. 351-358, 1995.

[20] Z. Zhang, “Iterative point matching for registration of free-form curves and surfaces,” *Int’l J. Computer Vision*, vol. 13, no. 2, pp. 119-152, 1994.