Toward a General 3-D Matching Engine: Multiple Models, Complex Scenes, and Efficient Data Filtering

Andrew Edie Johnson
Jet Propulsion Laboratory
California Institute of Technology
aej@robotics.jpl.nasa.gov

Owen Carmichael Daniel Huber Martial Hebert
The Robotics Institute
Carnegie Mellon University
{owenc,dhuber,hebert}@ri.cmu.edu

Abstract[1]

We present a 3-D shape-based object recognition system for simultaneous recognition of multiple objects in scenes containing clutter and occlusion. Recognition is based on matching surfaces by matching points using the spin-image representation. The spin-image is a data level shape descriptor that is used to match surfaces represented as surface meshes. Starting with the general matching framework introduced earlier, we present a compression scheme for spin-images; this scheme results in efficient multiple object recognition which we verify with results showing the simultaneous recognition of multiple objects from a library of 20 models. In addition, we demonstrate the robust performance of recognition in the presence of clutter and occlusion through analysis of recognition trials on 100 scenes. We address efficiency and generality through two extensions to the basic matching scheme: fast filtering of scene points and processing of general data sets.

1. Introduction

In 3-D object recognition, shape representations are used to collate the information conveyed by sensed surface points so that surfaces can be matched efficiently. For object recognition, the following criteria should be met for any shape representation used in realistic settings; the representation must represent general shapes, robust to clutter and occlusion, and be efficient.

In the past, the trend in object recognition has been to restrict the class of objects that can be recognized so that efficient matching schemes can be developed. (e.g., surface patches [6], super-quadrics [17] and spherical representations [3]). However, the real world comprises objects of many different shapes without regard to specific analytic surface descriptions. For an object recognition system to be widely applicable in the real world, the shape representation it uses cannot be restrictive.

Some of the more successful object recognition systems are designed to work on isolated and unoccluded objects (e.g., parametric appearance [14], COSMOS [5]) This is a serious disadvantage for these systems if the object is to be recognized in real scenes, because real scenes contain clutter and occlusion. Consequently, any recognition system designed to work in the real world must be robust to clutter and occlusion.

Our final requirement, efficiency, can be divided into two related requirements. Object representations should enable efficient matching of surfaces from multiple models, so that recognition occurs in a timely fashion. Furthermore, the representations should be efficient in storage (i.e., compact), so that many models can be stored in the model library. Without efficiency, a recognition system will not be able to recognize the multitude of objects that exist in the real world.

The approach taken in this paper is based on the spin-image representation introduced in [10]. Based on this representation, a general matching algorithm was also introduced in [10] and initial results for single object recognition and view registration were described. This matching algorithm, summarized in Section 2., is the only approach that is able to compare, match, and register arbitrary 3-D surfaces using point sets and meshes as the basic underlying representation. We present here extensions to the original technique for handling large model libraries (Section 3.), rapid filtering of scene data for computational efficiency, and using general point sets (Section 5.). We also report on experimental evaluation of the performance of the matching engine with respect to occlusion and clutter (Section 4.). The basic matching engine, together with the extensions, constitute a general 3-D matching toolbox based on a simple representation.

1. This research has been sponsored in full or in part by Office of Naval Research (ONR) under Contract N00014-95-1-0591. 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 ONR or the U.S.Government.

2. Surface matching

Our object representation is composed of two parts: a polygonal mesh describing the shape of the surface and a set of spin-images created from the surface mesh. Surface meshes are general shape representations because, given a sufficient number of vertices, surface meshes can represent almost any shape. This section provides the necessary background for understanding the rest of the paper; more complete descriptions of spin-images and our surface matching algorithms are given in [9] [10].

2.1. Spin-images

Oriented points, 3-D points with associated directions, are used to create spin-images. We define an oriented point at a surface mesh vertex using the 3-D position of the vertex and surface normal at the vertex. The surface normal at a vertex is computed by fitting a plane to the points connected to the vertex by edges in the surface mesh.

An oriented point defines a partial, object-centered, coordinate system. Two cylindrical coordinates can be defined with respect to an oriented point: the radial coordinate α, defined as the perpendicular distance to the line through the surface normal, and the elevation coordinate β, defined as the signed perpendicular distance to the tangent plane defined by vertex normal and position. The cylindrical angular coordinate is omitted because it cannot be defined robustly and unambiguously on planar surfaces.

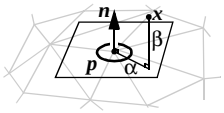

Figure 1. An oriented point basis defined at a surface mesh vertex.

A two-dimensional signature, called a spin-image, is created for an oriented point at a vertex in the surface mesh as follows. A 2-D accumulator indexed by α and β is created. Next, the coordinates (α,β) are computed for a vertex in the surface mesh that is within the support of the spin-image (explained below). The bin indexed by (α,β) in the accumulator is then incremented; bilinear interpolation is used to smooth the contribution of the vertex. This procedure is repeated for all vertices within the support of the spin-image. The resulting accumulator can be thought of as an image; dark areas in the image correspond to bins that contain many projected points. As long as the size of the bins in the accumulator is set on order of the median distance between vertices in the mesh [10], the position of individual vertices will be averaged out during spin-image generation.

Figure 2 shows the projected (α,β) coordinates and spin-images for three oriented points on a rubber duckie model. For surface matching, spin-images are constructed for every vertex in the surface mesh.

The support of a spin-image is the part of the surface of an object around the oriented point basis that can contribute points to spin-image generation. There are two parameters that control the spin-image support. The first parameter, *support distance* D_S, sets the maximum distance between the oriented point and a point contributing to the spin-image. Support distance is analogous to window size in 2-D template matching. The second parameter, *support angle* A_S, limits the angle between the normal of the oriented point basis and the normal of points contributing to the spin-image. As shown in Figure 3, by changing the support parameters, spin-images can be smoothly transformed from global to local representations. Spin-images of small support are robust to clutter and occlusion, while spin-images of large support are highly discriminating.

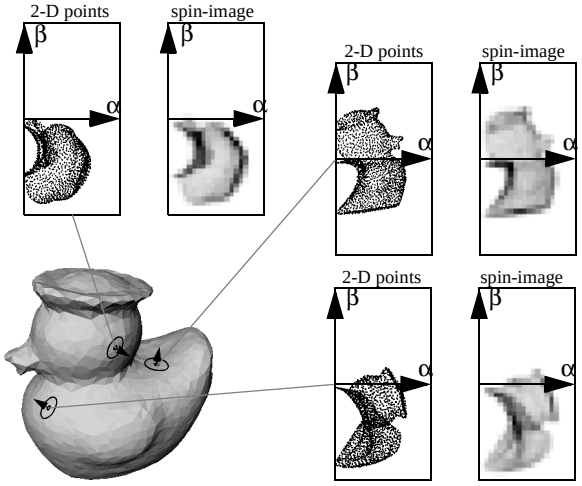

Figure 2. Spin-images of large support for three oriented points on the surface of a rubber duckie model.

Spin-images have some useful properties that distinguish them from other representations for 3-D surface matching. Since spin-images are object centered representations, they are invariant to rigid transformations. Consequently, comparison of spin-images can be used to establish point correspondences between different views of the same object. Spin-images can represent general shapes because they are constructed directly from polygonal surface meshes without surface fitting (except for surface normal). Finally, since spin-images are image-based descriptions of 3-D shape, many of the powerful tools available for 2-D image analysis can also be used to analyze spin-images.

2.2. Surface matching engine

Two surfaces are matched as follows. Spin-images from points on one surface are compared by computing the correlation coefficient with spin-images

from points on another surface; when two spin-images are highly correlated, a point correspondence between the surfaces is established. Point correspondences are then grouped and outliers are eliminated using geometric consistency. Groups of geometrically consistent correspondences are then used to calculate a rigid transformation that aligns one surface with the other. After alignment, surface matches are verified using a modified iterative closest point algorithm. Details of the surface matching engine are given in [9].

Various factors, including surface normal noise and the symmetry in spin-image generation, can cause the generation of similar spin-images for points that should not be matched. It would appear that these factors would prevent the matching of surfaces using spin-images. However, a detailed analysis of spin-image matching and results on real scenes have shown that spin-images maintain their descriptiveness even in the presence of large sensor noise, clutter, occlusion, and local object symmetry [10].

3. Object recognition

Surface matching using spin-images can be extended to object recognition as follows. Each model in the model library is represented as a polygonal mesh. Before recognition, the spin-images for all vertices on all models are created and stored. At recognition time, a scene point is selected and its spin-image is generated. Then that spin-image is correlated with all of the spin-images from all of the models. The best matching model spin-image will indicate both the best matching model and model vertex. After matching many scene spin-images to model spin-images, the point correspondences are input into the surface matching engine described in Section 2.2.. The result is simultaneous recognition and localization of the models that exist in the scene.

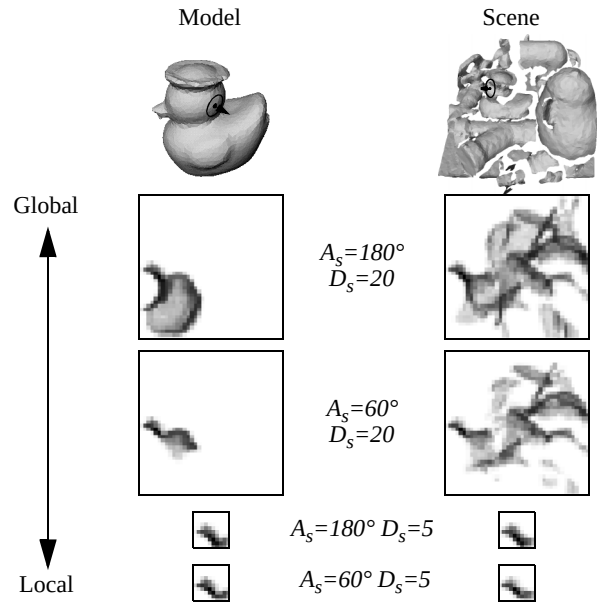

Figure 3. How spin-image generation parameters localize spin-images to reduce the effect of scene clutter and occlusion on matching.

This form of surface matching is inefficient for two reasons. First, each spin-image comparison requires a correlation of two spin-images, an operation on order of the relatively large (~200) number of bins in a spin-image. Second, when a spin-image is matched to the model library, it is correlated with all of the spin-images from all of the models. This operation is linear in the number of vertices in each model and linear in the number of models. This linearly growth rate is unacceptable for recognition from large model libraries. Fortunately, spin-images can be compressed to considerably speed up matching.

3.1. Spin-image compression

Spin-images coming from the same surface can be correlated for two reasons: First, spin-images generated from oriented point bases that are close to each other on the surface will be correlated. Second, surface symmetry and the inherit symmetry of spin-image generation will cause two oriented point bases on equal but opposite sides of a plane of symmetry to be correlated. Furthermore, surfaces from different objects can be similar on the local scale. Therefore, there can exist a correlation between spin-images of small support generated for different objects.

This correlation can be exploited to make spin-image matching more efficient through image compression. For compression, it is convenient to think of spin-images as vectors in a D-dimensional vector space where D is the number of pixels in the spin-image. Correlation between spin-images places the set of spin-images in a low dimensional subspace of this D-dimensional space.

A common technique for image compression in object recognition is principal component analysis (PCA) [14]. PCA or Karhunen-Loeve expansion [7] is a well known method for computing the directions of greatest variance for a set of vectors. By computing the eigenvectors of the covariance matrix of the set of vectors, PCA determines an orthogonal basis, called the eigenspace, in which to describe the vectors. PCA has become popular for efficient comparison of images because it is optimal in the correlation sense [7].

PCA is used to compress the spin-images coming from all models simultaneously as follows. Suppose the model library contains N spin-images $\boldsymbol{x_i}$ of size D. First, to make the principal directions computed by PCA more effective for describing the

variance between spin-images, the mean of all spin-images X is subtracted from each spin-image.

$$\hat{x}_i = x_i - \bar{x} \quad (1)$$

The set of mean-subtracted spin-images can be represented as an *DxN* matrix, with each column of the matrix being a mean-subtracted spin-image

$$S^m = \left[\hat{x}_1\ \hat{x}_2\ \ldots\ \hat{x}_N\right]. \quad (2)$$

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

$$C^m = S^m (S^m)^T. \quad (3)$$

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

$$\lambda_i^m e_i^m = C^m e_i^m. \quad (4)$$

Since the dimension of the spin-images is not too large (~200), the standard **`jacobi`** algorithm from Numerical Recipes in C [16] is used to determine the eigenvectors e_j^m and eigenvalues λ_j^m of C^m. Since the eigenvectors of C^m can be considered spin-images, they are called eigen-spin-images.

Next the model projection dimension, *s*, is determined using a reconstruction metric that depends on the needed fidelity in reconstruction and the variance among images (see [10]). Every spin-image from each model is then projected onto the *s*-dimensional subspace spanned by the *s* eigenvectors of largest eigenvalue; the s-tuple of projection coefficients, p_j, becomes the compressed representation of the spin-image.

$$p_j = (\hat{x}_j e_1^m, \hat{x}_j e_2^m, \ldots, \hat{x}_j e_s^m) \quad (5)$$

The amount of compression is defined by *s/D*. The compressed representation of a model library has two components: the *s* most significant eigenvectors and the set of *s*-tuples, one for each model spin-image. Since the similarity between images is determined by computing the l_2 distance between *s*-tuples, the amount of storage for spin-images and the time required to compare them is reduced.

3.2. Matching compressed spin-images

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

To match a scene spin-image to a model *s*-tuple, a scene *s*-tuple must be generated as follows. First the scene spin-image is generated. Next the mean of the library spin-images is subtracted from the scene spin-image. Finally, the scene spin-image is projected onto the top *s* library eigen-spin-images to get the scene *s*-tuple

To determine the best matching model spin-image to scene spin-image, the l_2 distance between the scene and model tuples is used. When comparing compressed model spin-images, finding closest *s*-tuples replaces correlating spin-images. Although the l_2 distance between spin-images is not the same as the correlation coefficient used in spin-image matching (correlation is really the normalized dot product of two vectors), it is still a good measure of the similarity of two spin-images.

To find closest points, we use the efficient closest point search structure proposed by Nene and Nayar [15]. The efficiency of their data structure is based on the assumption that one is interested only in the closest point, if it is less than a predetermined distance ε from the query point. This assumption is reasonable in the context of spin-image matching, so we chose their data structure. Furthermore, in our experimental comparison, we found that using their data structure resulted in order of magnitude improvement in matching speed over matching using kd-trees or exhaustive search. The applicability of the algorithm to the problem of matching *s*-tuples is not surprising; the authors of the algorithm demonstrated its effectiveness in the domain of appearance-based recognition [14], a domain that is similar to spin-image matching. In out implementation, the search parameter ε was automatically set to the average of the closest distances between model *s*-tuples. Setting ε in this way balances the likelihood of finding closest points against closest point lookup time.

Spin-image matching with compression is very similar to the recognition algorithm without compression. Before recognition, all of the model surface meshes are resampled to the same resolution to avoid scale problems when comparing spin-images from different models. Next, the spin-images for each model in the model library are generated, and the library eigen-spin-images are computed. The projection dimension *s* is then determined for the library. Next, the *s*-tuples for the spin-images in each model are computed by projecting model spin-images onto library eigen-spin-images. Finally, model *s*-tuples are stored in the efficient closest point search structure.

At recognition time, a fraction of oriented points are selected at random from the scene. For each scene oriented point, a scene *s*-tuple is computed. The scene *s*-tuple is then used as a query point into the library efficient closest point search structure which returns a list of model *s*-tuples close to the scene *s*-tuple. These point matches, which encode model and model vertex, are then fed into the surface matching engine to find model/scene surface matches.

3.3. Results

To test our recognition system, we created a model library containing twenty complete object models. The models in the library are shown in Figure 4; each was created by registering and integrating multiple range views of the objects. Next, cluttered scenes were created by pushing objects into a pile and acquiring a range image with a K^2T structured light range finder. The scene data was then processed to remove faces on occluding edges, isolated points, dangling edges and small patches. This topological filter was followed by mesh smoothing without shrinking and mesh simplification [10] to change the scene data resolution to that of the models in the model library.

Figure 5 shows the simultaneous recognition of seven models from the library of twenty models with a compression ratio of $s/D = 0.1$. In the top right of the figure is shown the intensity image of the scene, and in the top left is shown the scene intensity image with the position of recognized models superimposed as white dots. In the middle is shown a frontal 3-D view of the scene data, shown as wireframe mesh, and then the same view of the scene data with models superimposed as shaded surfaces. The bottom shows a top view of the scene and models. From the three views it is clear that the models are closely packed a condition which creates a cluttered scene with occlusions.

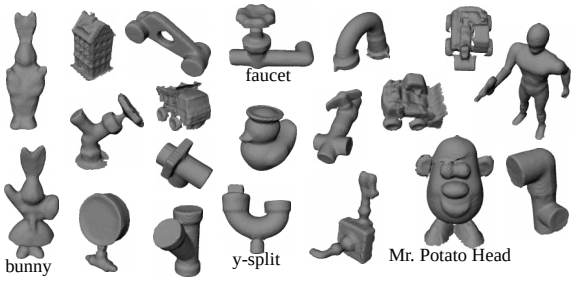

Figure 4. 20 Models used for recognition.

Because spin-image matching has been designed to be resistant to clutter and occlusion, our algorithm is able to recognize the seven most prominent objects in the scene with no incorrect recognitions. Some of the objects present were not recognized because insufficient surface data was present for matching. Figure 6 shows the recognition of 6 objects in a similar format to Figure 5.

4. Recognition in complex scenes

Any recognition algorithm designed for the real world must work in the presence of clutter and occlusion. In Section 2.1., we claim that creating spin-images of small support will make our representation robust to clutter and occlusion. In this section, this claim is verified experimentally. We have developed an experiment to test the effectiveness of our algorithm in the presence of clutter and occlusion.

Stated succinctly, the experiment consists of acquiring many scene data sets, running our recognition algorithms on the scenes, and then interactively measuring the clutter and occlusion in each scene along with the recognition success or failure. By plotting recognition success or failure against the amount of clutter or occlusion in the scene, the effect of clutter and occlusion on recognition can be determined.

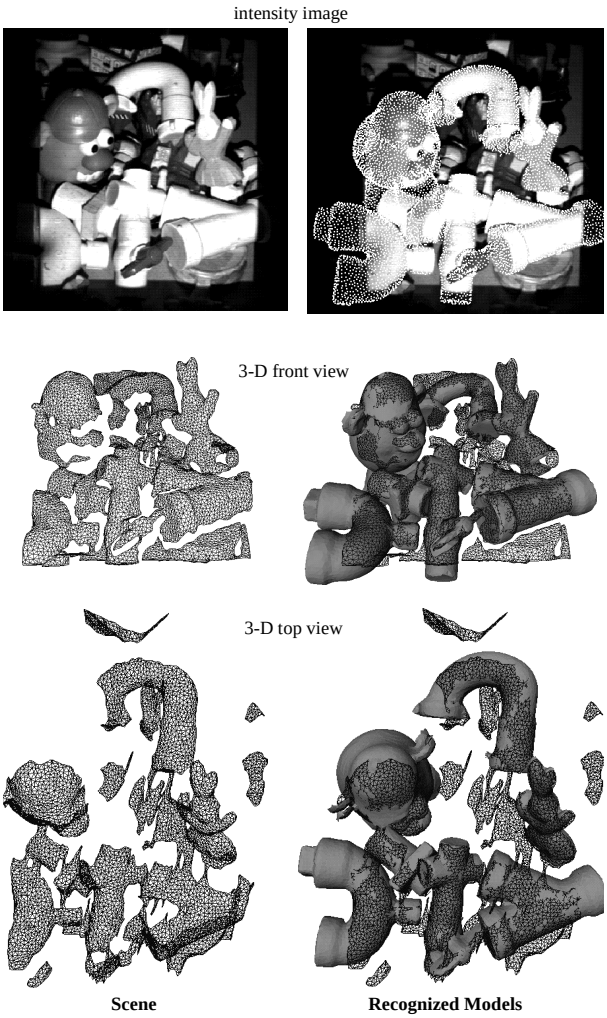

Figure 5. Simultaneous recognition of 7 models from a library of 20 models in a cluttered scene.

4.1. Experiments

Recognition success or failure can be broken down into four possible recognition states. If the model exists in the scene and is recognized by the algorithm, this is termed a *true-positive* state. If the model does not exist in the scene, and the recognition algorithm concludes that the model does exist in the scene or places the model in an entirely incorrect position in the scene, this is termed a

false-positive state. If the recognition algorithm concludes that the model does not exist in the scene when it actually does exist in the scene, this is termed a *false-negative* state. The *true-negative* state did not exist in our experiments because the model being searched for was always present in the scene.

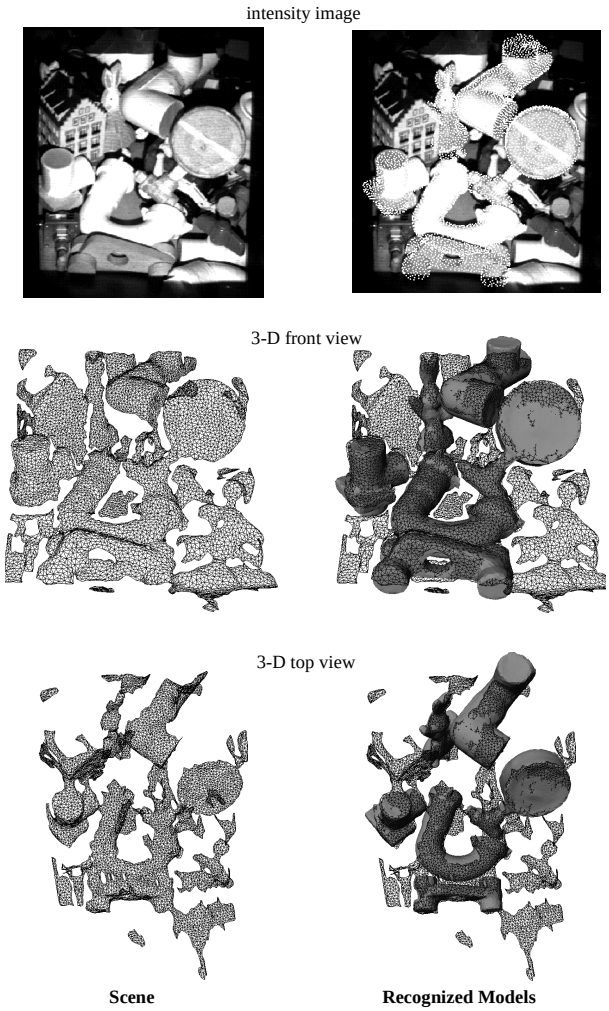

Figure 6. Simultaneous recognition of 6 models from a library of 20 models in a cluttered scene.

In our experiment for measuring the effect of clutter and occlusion on recognition, a *recognition trial* consists of the following steps. First, a model is placed in the scene with some other objects. The other objects might occlude the model and will produce scene clutter. Next, the scene is imaged and the scene data is processed as described in Section 3.3. A recognition algorithm that matches the model to the scene data is applied and the result of the algorithm is presented to the user. Using a graphical interface, the user then interactively segments the surface patch that belongs to the model from the rest of the surface data in the scene. Given this segmentation, the amounts of clutter and occlusion are automatically calculated as explained below. By viewing the model superimposed on the scene, the user decides the recognition state; this state is then recorded with the computed clutter and occlusion. By executing many recognition trials using different models and many different scenes, a distribution of recognition state versus the amount of clutter and occlusion in the scene is generated.

The occlusion of a model is defined as

$$\text{occlusion} = 1 - \frac{\text{model surface patch area}}{\text{total model surface area}} \qquad (6)$$

Surface area for a mesh is calculated as the sum of the areas of the faces making up the mesh. The clutter in the scene is defined as

$$\text{clutter} = \frac{\text{clutter points in relevant volume}}{\text{total points in relevant volume}} \qquad (7)$$

Clutter points are vertices in the scene surface mesh that are not on the model surface patch. The relevant volume is the union of the supports (Section 2.1.) of all of the oriented points on the model surface patch. If the relevant volume contains points that are not on the model surface patch, then these points will corrupt scene spin-images and are considered clutter points.

We created 100 scenes for analysis as follows. We selected four models from our library of models: bunny, faucet, Mr. Potato Head and y-split (Figure 4). We then created 100 scenes using these four models; each scene contained all four models. The models were placed in the scenes without any systematic method. It was our hope that random placement would result in a uniform sampling of all possible scenes containing the four objects.

4.2. Analysis

For each model, we ran recognition without compression on each of the 100 scenes, resulting in 400 recognition trials.The recognition states are shown in a scatter plot in the top left of Figure 7. Each data point in the plot corresponds to a single recognition trial; the coordinates give the amount of clutter and occlusion and the symbol describes the recognition state. This same procedure using the same 100 scenes was repeated for the matching spin-images with compression (*s/D = 0.1)* resulting in 400 different recognition runs. A scatter plot of recognition states for compressed spin-images is shown at the bottom left of Figure 7. A brief examination of both scatter plots shows that the number of true-positive states is much larger than the number of false negative states and false-positive states. Furthermore, as the lines in the scatter plots indicate, no recognition errors occur below a fixed level of occlusion, independent of the amount of clutter.

Examining the scatter plots in Figure 7, one notices that recognition rate is affected by occlusion. At low occlusion values, no recognition failures are reported, while at high occlusion values, recogni-

tion failures dominate. This indicates that recognition will almost always work if sufficient model surface area is visible. The decrease in recognition success after a fixed level of occlusion is reached (70%) indicates that spin-image matching does not work well when only a small portion of the model is visible. This is no surprise since spin-image descriptiveness comes from accumulation of surface area around a point. In the middle of Figure 7 are shown the experimental recognition rates versus scene occlusion. The rates are computed using a gaussian weighted running average to avoid the problems with binning. These plots show that recognition rate remains high for both forms of compression until occlusion of around 70% is reached, then the successful recognition rate begins to fall off.

Examining the experiment scatter plots in Figure 7, one notices that the effect of clutter on recognition is uniform across all levels of occlusion until a high level of clutter is reached. This indicates that spin-image matching is independent of the clutter in the scene. On the right in Figure 7, plots of recognition rate versus amount of clutter also show that recognition rate is fairly independent of clutter. As clutter increases, there are slight variations about a fixed recognition rate. Most likely, these variations are due to non-uniform sampling of recognition runs and are not actual trends with respect to clutter. Above a high level of clutter, the successful recognitions decline, but from the scatter plots we see that, at high levels of clutter, the number of experiments is small, so conclusions about recognition rate can not be made.

In all of the plots showing the effect of clutter and occlusion, the true-positive rates are higher for recognition with spin-images without compression when compared with the true-positive rates for recognition with compression. This validates the expected decrease in the accuracy of spin-image matching when using compressed spin-images. However, it should be noted that the recognition rate for both matching algorithms remains high. For all levels of clutter and occlusion, matching without compression has an average recognition rate of 90.0% and matching with compression has an average recognition rate of 83.2%. Furthermore, the false-positives rates for both algorithms are low and nearly the same. Our experiments show that the decrease in recognition rate for matching with compression is compensated for by an order of magnitude increase in matching speed [10].

5. Extensions

Two additional extensions to the basic are of particular interest. First, enhancements in computational efficiency are needed in order to be able to cope with very large data sets and complex scenes. In particular, it is often the case that the model occupies only a tiny fraction of the observed scene, in which case most of the matching is wasted filtering out points that are far from the model shape. Second, the input data may not be have uniformly high density. There may be area of sparse data mixed with areas of dense data in the scene. There fore, the matching algorithm must be designed to handle data sets as general as possible. Those

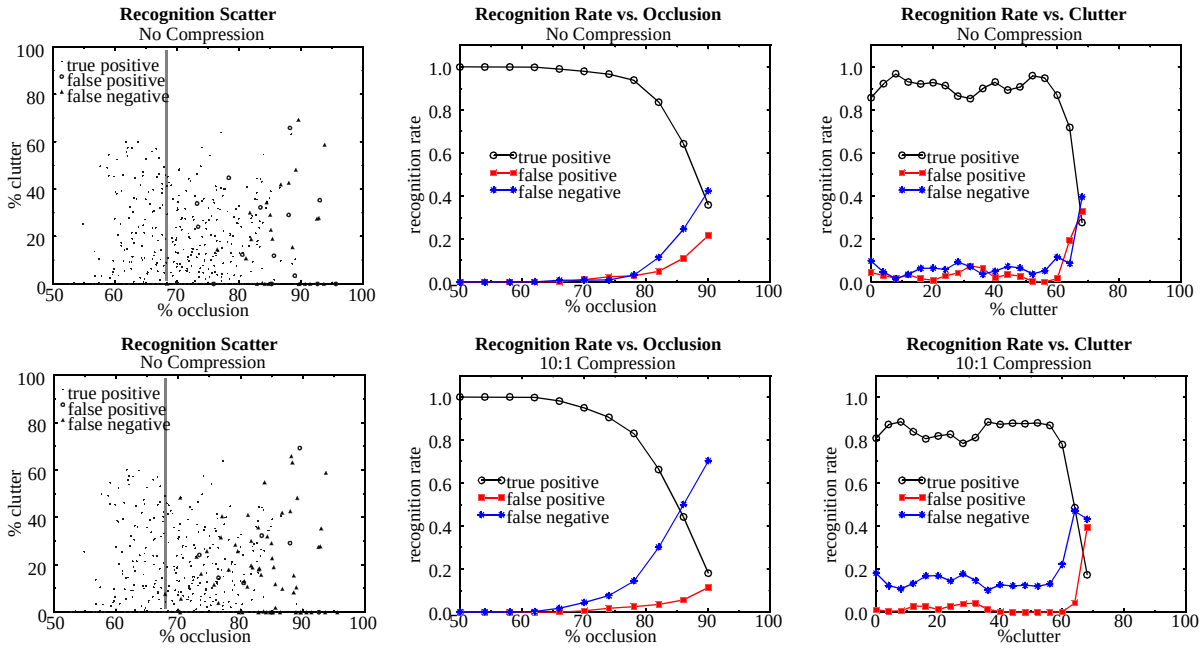

Figure 7. Recognition states vs. clutter and occlusion for compressed and uncompressed spin-images (left). Recognition state probability vs. occlusion for compressed and uncompressed spin-images (middle). Recognition state probability vs. clutter (right).

enhancements are described in detail in the next two sections.

5.1. Cueing

In the matching approach described so far, points are randomly selected in the scene as basis points for comparison with the basis points of the model. This works well so long as the points are selected using a uniform distribution. An undesirable result of this approach is that, in scene in which the object of interest occupies only a small percentage of the visible surface area, most of the scene points that are evaluated for matching are rejected. In other words, the matcher may spend a lot of time testing points that are nowhere close to the object of interest. An obvious enhancement to the matching algorithm would be a mechanism by which a large percentage of the points in the scene could be filtered out. We call this mechanism "cueing," in that it suggests possible areas of the scene where the object might be located. The matching engine can confirm or reject points in the suggested areas. Matching can benefit from such a cueing mechanism only if it is simple and fast compared with the matching algorithm.

The approach chosen here is to combine all the spin images from the model into a combined model which can be quickly compared with each signature image from the scene. This approach is similar to the generalization problems in machine learning. Specifically, the method of naive Bayesian classification was chosen as a means to determine which scene points have high probability of being on a model.

Let X_i be the spin image associated with point i in the scene, and let $i \rightarrow A$ denote the hypothesis that scene point i lies on object A. Then by Bayes rule,

$$P(i \rightarrow A | X_i) = \frac{P(X_i | i \rightarrow A) P(i \rightarrow A)}{P(X_i)}$$

To find scene points which have the highest probability of belonging to a model, we need to find the MAP hypotheses $i \rightarrow A$; that is, the point i which maximize $P\langle i \rightarrow A | X_i \rangle$. No data is known about the prior probability that any particular point i may be found on a model, and no data is known a priori about the probability of a particular spin image X_i occurring, so uniform distributions $P\langle i \rightarrow A \rangle$ and $P\langle X_i \rangle$ are assumed. This reduces the problem to finding $max_i \; P(X_i | i \rightarrow A)$ or the maximum-likelihood hypothesis of $i \rightarrow A$ given X_i over all i's. It would not be feasible, however, to compute $P(X_i | i \rightarrow A)$ because two spin images taken from different points on the same object or different objects almost never have exactly identical spin images; even spin images for corresponding points will differ slightly from pixel to pixel, causing $P(X_i | i \rightarrow A)$ to almost always evaluate to 0. Because of this, we assume conditional independence between the pixels in X_i [4]; that is, if $X_i(\alpha,\beta)$ denotes the value of bin (α,β) in X_i, then we assume

$$P(X_i | i \rightarrow A) = \prod_{m,\, n} P(X_i(m, n) | i \rightarrow A)$$

where $P(X_i(m, n) | i \rightarrow A)$ is the probability that $X_i(\alpha,\beta)$ for a given spin image X_i will equal what it does given that point i is on model A. This assumption of conditional independence of spin image pixels thus reduces this method to naive Bayesian classification.

The distribution for $P\langle X_i(\alpha,\beta) | i \rightarrow A \rangle$ is discretized into bins, usually 10 or 20. During training, probabilities $P\langle X_i(\alpha,\beta) | i \rightarrow A \rangle$ are first estimated from the spin images of scene points; for every model spin image X_i, for every pixel (α,β), $P\langle X_i(\alpha,\beta) | i \rightarrow A \rangle$ is incremented, and at the end, each $P\langle X_i(\alpha,\beta) | i \rightarrow A \rangle$ is divided by the number of images that contributed to it. Then, at run time, spin images are constructed for scene points, and the above product estimating $P\langle X_i | i \rightarrow A \rangle$ is found by looking up $P\langle X_i(\alpha,\beta) | i \rightarrow A \rangle$ for every pixel (α,β) in the scene spin image. Those scene spin images with maximum $P\langle X_i | i \rightarrow A \rangle$ are considered most likely to be members of a model.

This method of point classification was tested on scene images containing a variety of cluttered scenes. Performance of the classifier is evaluated by taking the ratio of the percentage of points correctly classified by the classifier to the percentage of the scene that really consisted of model points. Formally, if N_m is the number of model points in the scene according to ground truth, N_c is the number of clutter points in the scene, N'_m is the number of scene points that were classified as model points and actually were points on the model, and N_c is the number of scene points classified as model points, but were really points lying somewhere on the clutter, then we define the performance of the classifier by the ratio $r = \rho'/\rho$, where $\rho' = N'_m/(N'_c + N'_m)$ and $\rho = N_m/(N_c + N_m)$. A high value for r indicates a reduction in the number of points that are considered for matching. If the filtering were perfect, all the points would lie on the model, therefore, $r = 1/\rho$ is the maximum value of r. Figure 8. shows the performance of the classifier on a test set of scenes. The results show that the classifier performed better than random selection ($\rho = 1$) in all cases, and often close to the optimal ($r = 1/\rho$.) Obviously, the performance of the classifier degrades in scenes containing multiple instances of very similar surfaces. However, it never makes the initial random selection worse, i.e., ρ' is always greater than ρ. Figure 9. shows two examples of object cueing using an elbow joint as the model. The points selected by the classifier are shown as black dots. Most of the points are correctly selected on the object in this example.

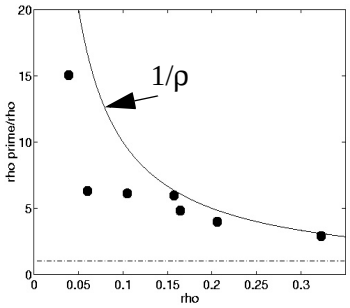

Figure 8. Cueing performance.

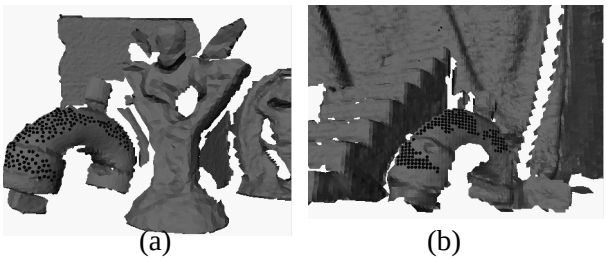

Figure 9. Cueing examples.

5.2. Surface Interpolation

In the representation described thus far, the signature images are computed by histogramming the vertices of the model or scene meshes. As a result, the distribution of the vertices on the mesh directly affects the spin-images. In fact, two meshes with different vertex distributions may generate very different signatures at the same basis point. Therefore, in order for the surface matching algorithm to work properly, some constraint has to be enforced on the distribution of vertices on the meshes. Specifically, it can be shown that the spin-images remain stable as long as the vertices are uniformly distributed on the surface. In all the results presented thus far, a decimation algorithm was applied to all the meshes prior to matching in order to enforce this uniformity constraint. The decimation algorithm is described in detail in [11].

Although this approach works well in practice, it has several problems. First of all, there are cases in which the data simply cannot be made uniform without loosing a great deal information because the variation of resolution in the input sensor data is too large. A typical example is terrain data taken from a forward-looking sensor. The sensor data varies from high-resolution at close range to quadratically decreasing resolution as the range from the sensor increases. Variations in data point spacing of as much as 1:10 are routinely observed on terrain data. The second problem is that the uniform decimation requires on the same order of computation time as the matching itself, even though much faster decimation and filtering algorithms do exist [8]. Finally, as a guiding principle of this work, we aim to make the matching algorithm as general as possible. In that respect, using arbitrary mesh distributions is critical.

The solution to those problems is to compute the signatures by integrating over the entire surface rather than by computing α and β values at the vertices only. Essentially, this requires interpolating the spin-image values "in between" the mesh vertices. The simplest way of achieving this is to raster scan each triangle of the mesh (Figure 10) and to compute the (α,β) coordinates of each point inside the face. The corresponding spin-image entry is incremented by a constant amount for each new point. This algorithm can be made efficient by using a fast geometric test in order to determine whether a face is inside the region of influence of the basis point and is within the boundary of the spin-image space.

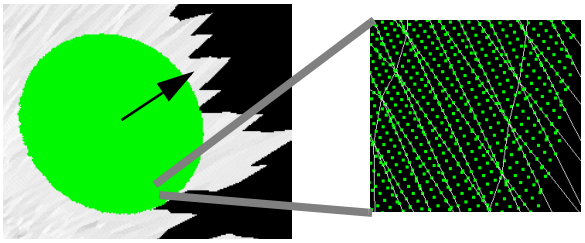

Figure 10. Discrete vertex interpolation: each face is raster-scanned before mapping to spin-image.

This approach is still an approximation because it uses a discrete sampling of the surface. In particular, although the signatures are less sensitive to the distribution of vertices, they are still sensitive to the choice of the sampling rate used for interpolation.

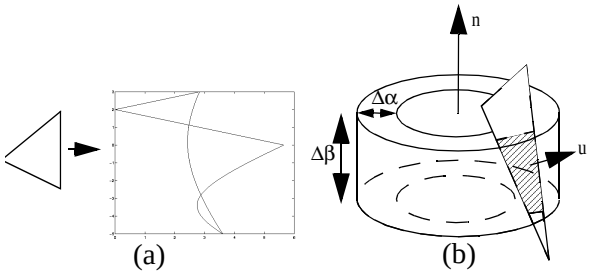

Figure 11. Continuous interpolation: each face is mapped to $\alpha\beta$ space by mapping its edges (a); cells inside the mapped region are incremented by the area of the intersection of the face and the volume in space corresponding to the cell (b).

The second approach is exact in that it computes the spin-images by integration over the whole surface without additional sampling. In this approach, the boundary of each triangle is mapped into $\alpha\beta$-space, as shown in Figure 11 (a). Each edge of the face maps to a segment of hyperbola. The hyperbolic segments computed in the projection are then

used for determining which cells of the spin-image may contain some portion of the triangle. Figure 11 (b) shows the portion of the spin-image that contains a portion of the triangle based on the segments of Figure 11 (a). Finally, each cell in the spin-image is incremented by the area of the part of the triangular face that is mapped to that cell in $\alpha\beta$-space.

This last step is illustrated geometrically in Figure 11 (c). The region of 3D corresponding to the cell (α,β) is an annulus of height $\Delta\beta$ and thickness $\Delta\alpha$. The cell is incremented by the surface area of the intersection between the triangle and this annulus. Calculating this intersection can be made efficient by observing that the 2-D intersection *a* in a plane perpendicular to the surface normal at the basis point can be computed first. The actual area used for histogram accumulation is $A = a/n.v.$

Because it uses the actual surface area for incrementing the spin-image cells, this algorithm computes an "exact" mapping of the surface to the signatures, given a mesh discretization of the surface. A systematic comparison of the discrete and continuous approaches remains to be done, but the benefits of interpolation over vertex mapping have been demonstrated. Figure 12 shows two meshes of an object at two different resolutions and different point distributions. Given a basis point, the region of influence of which is shown on the mesh, the signatures computed from the only vertices are quite different as expected. However, the signatures computed using interpolation are similar.

How much more similar are the signatures computed using interpolation? A partial answer to that question is shown in Figure 14. An object (a faucet) was discretized at three different resolutions, denoted by *r1*, *r2*, and *r3* using the algorithm of [8], i.e., no attempt was made to enforce uniformity of the point distribution. The number of vertices on the three versions of the object is: 1585, 57, and 120, respectively. A set of points $\{p_i\}$ was selected on the object independently of the discretization and, for each basis point, the similarity of the signatures $s(r, r', p_i)$ was computed using different pairs of resolutions $r, r' = r1, r2$, and $r3$. The similarity is computed using the formula introduced in [10] and is close to zero for uncorrelated spin-images and has high values for similar images. Figure 14. shows the histograms of $s(r1, r2, p_i)$ and $s(r1, r3, p_i)$ using the vertex method and the interpolation method. The histograms show a substantial improvement in the similarity of signatures. In particular, most of the similarity values $s(r1, r3, p_i)$ are so low that such a difference in resolution makes matching impractical. Using the interpolation method, however, increases the similarity values to a level suitable for matching.

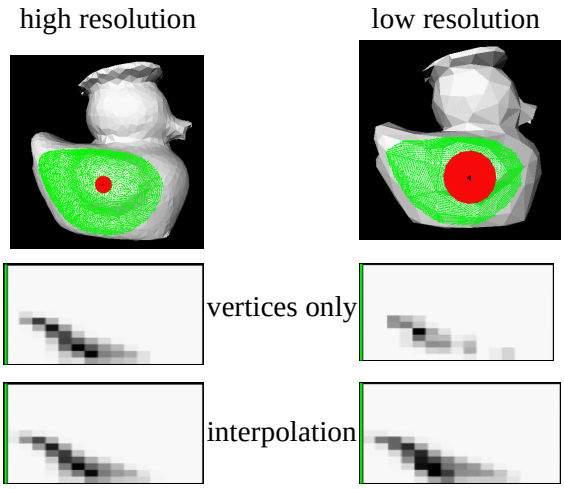

Figure 12. Comparison of the signatures at a basis point computed at different resolutions using the vertex and interpolation methods.

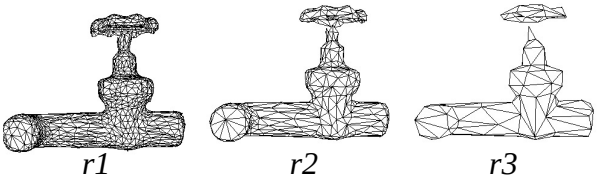

Figure 13. Test object for interpolation at three different resolutions.

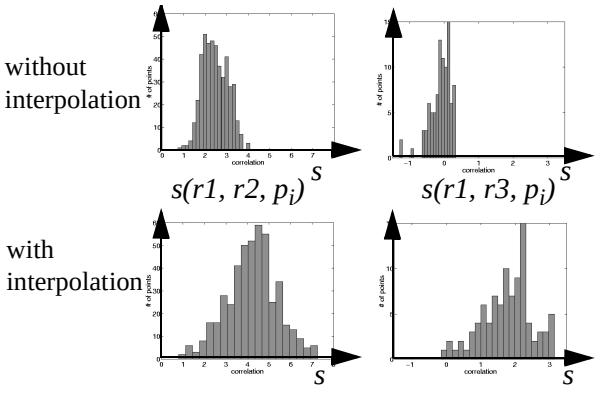

Figure 14. Histograms of similarity between meshes at different resolutions with and without interpolation.

This experiment shows that, by performing integration over the surface rather than resolution-dependent sampling, the interpolation method permits the comparison of surfaces at very different levels of resolution. The examples of Figure 12 and Figure 14 were obtained from controlled experiments; the question of the effectiveness of surface interpolation remains. Figure 15 shows a matching example in which two large terrain maps are matched and registered. The maps extend over 50m radially over a 180^o sector. The original was

computed from 300x600 images and simplified to 10,000 faces for the purpose of matching. In this example, the mesh resolution in the simplified mesh varies from 9m to 2cm. Because of this large variation in the distribution of vertices, the surfaces could not be matched without using the interpolation algorithm.

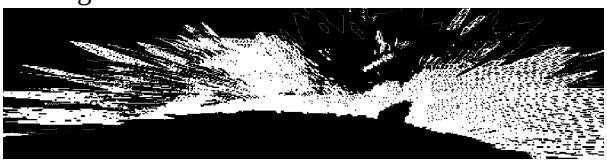

Figure 15. Result of registering two surfaces with uneven vertex distribution using the interpolation method.

6. Conclusion

We have presented an algorithm for simultaneous shape-based recognition of multiple objects in cluttered scenes with occlusion. Because it is based on the spin-image representation, our algorithm can handle objects of general shape. A data level shape representation that places few restrictions on object shape. Through compression of spin-images using PCA, we have made the spin-image representation efficient enough for recognition from large model libraries. We have shown experimentally, that the spin-image representation is robust to clutter and occlusion. Through improvements and analysis, we have shown that the spin-image representation is an appropriate representation for recognizing objects in complicated real scenes.

We have presented two extensions to the basic matching approach. Cueing provides a simple and effective way to extract the most promising candidate points for matching out of arbitrarily complex scenes. This extension permits the use of surface matching for large model library and complex scenes. A second extension, vertex interpolation, eliminates additional constraints on the input data. It permits surface matching and recognition even in cases when the distribution of the data points on the surfaces is uneven and in cases in which models and scenes are represented by point sets of substantially different distributions.

Together, the surface matching engine and its extensions provide a toolbox of techniques directly applicable to a number of problems. Examples of applications to object recognition of the surface matching engine can be found in [12]; applications to mapping can be found in [2].

References

[1] C. Chua and R. Jarvis. 3-D free-form surface registration and object recognition. *Int'l Jour. Computer Vision*, vol. 17, no. 1, pp. 77-99, 1996.

[2] O. Carmichael and M. Hebert. Unconstrained Registration of Large 3D Point Sets for Complex Model Building. *IEEE/RSJ International Conference On Intelligent Robotic Systems.* Vancouver. 1998.

[3] H. Dellingette, M. Hebert and K. Ikeuchi. A spherical representation for the recognition of curved objects. *Proc. Computer Vision and Pattern Recognition (CVPR '93*), pp. 103-112, 1993.

[4] P. Domingos and M. Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. Proc. 13th Int'l Conference of Machine Learning (pp. 105-112). 1996.

[5] C. Dorai and A. Jain. COSMOS - A representation scheme for 3D free-form objects. *Pattern Analysis and Machine Intelligence*, vol. 19, no. 10, pp. 1115-1130, 1997.

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

[7] K. Fukanaga. *Introduction to Statistical Pattern Recognition*. Academic Press, New York, 1972.

[8] M. Garland and P. S. Heckbert. Surface Simplification Using Quadric Error Metrics. SIGGRAPH 97.

[9] A. Johnson and M. Hebert. Object recognition by matching oriented points. *Proc. Computer Vision and Pattern Recognition (CVPR '97)*, pp. 684-689, May 1997.

[10] A. Johnson. *Spin-images: A Representation for 3-D Surface Matching.* Ph.D. Thesis, The Robotics Institute, Carnegie Mellon University, November 1997.

[11] A. Johnson and M. Hebert. *Control of Polygonal Mesh Resolution for 3-D Computer Vision.* Graphics, Modeling, and Computer Vision.1998.

[12] A. Johnson, R. Hoffman, J. Osborn, M. Hebert. A System for Semi-Automated Modeling of Complex Environments. *Proc. International Conference on Recent Advances in 3-D Digital Imaging and Modeling*. Ottawa, May 1997.

[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] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int'l Jour. Computer Vision*, vol. 14, pp. 5-24, 1995.

[15] S. Nene and S Nayar. Closest point search in high dimensions. *Proc. Computer Vision and Pattern Recognition* (CVPR '96), 1996.

[16] W. Press, S. Teukolsky, W. Vetterling and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, 2nd Ed.* Cambridge University Press, Cambridge, UK, 1992.

[17] N. Raja and A. Jain. Recognizing geons from superquadrics fitted to range data. *Image and Vision Computing*, vol. 10, no. 3, pp. 179-190, 1992.

[18] 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.