Using Spin-Images for Efficient Object Recognition in Cluttered 3-D Scenes

Andrew E. Johnson

Jet Propulsion Laboratory
Mail Stop 107-102
4800 Oak Grove Drive
Pasadena, CA 91109
aej@robotics.jpl.nasa.gov

Martial Hebert

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
hebert@ri.cmu.edu

Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence
submitted: July 1998

Abstract

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. We present a compression scheme for spin-images that results in efficient multiple object recognition which we verify with results showing the simultaneous recognition of multiple objects from a library of 20 models. Furthermore, we demonstrate the robust performance of recognition in the presence of clutter and occlusion through analysis of recognition trials on 100 scenes.

This research was performed at Carnegie Mellon University and was supported by the US Department of Energy under contract DE-AC21-92MC29104.

1 Introduction

Surface matching is a technique from 3-D computer vision that has many applications in the area of robotics and automation. Through surface matching, an object can be recognized in a scene by comparing a sensed surface to an object surface stored in memory. When the object surface is matched to the scene surface, an association is made between something known (the object) and something unknown (the scene); information about the world is obtained. Another application of surface matching is the alignment of two surfaces represented in different coordinate systems. By aligning the surfaces, the transformation between the surface coordinate systems is determined. Surface alignment has numerous applications including localization for robot navigation [29] and modeling of complex scenes from multiple views [15].

Shape representations are used to collate the information stored in sensed 3-D points so that surfaces can be compared efficiently. Shape can be represented in many different ways, and finding an appropriate representation for shape that is amenable to surface matching is still an open research issue[12]. The variation among shape representations spans many different axes. For instance, shape representations can be classified by the number of parameters used to describe each primitive in the representation. Representing objects using planar surface patches [8] uses many primitives, each with a few parameters. On the other hand, representing an object with generalized cylinders [6] requires fewer primitives, but each has many parameters. Another axis of comparison is the local versus global nature of the representation. The gaussian image [19] and related spherical representations [11] are global representations useful for describing single objects, while surface curvature [13] measures local surface properties and can be used for surface matching in complex scenes. The multitude of proposed surface representations indicates the lack of consensus on the best representation for surface matching.

Another factor determining the appropriate surface representation is the coordinate system in which the data is described. Surfaces can be defined in viewer-centered coordinate systems or object-centered coordinate systems. Viewer-centered representations [7] describe surface data with respect to a coordinate system dependent on the view of the surface. Although viewer-centered coordinate systems are easy to construct, the description of the surface changes as viewpoint changes, and surfaces must be aligned before they can be compared. Furthermore, to represent a surface from multiple views, a separate representation must be stored for each different viewpoint.

An object-centered coordinate system describes an object surface in a coordinate system fixed to the object. In object-centered coordinates, the description of the surface is view-independent, so surfaces can be directly compared, without first aligning the surfaces. Object-centered representations can be more compact than viewer-centered representations because a single surface representation describes all views of the surface. Finding an object-centered coordinate system is difficult because these systems are generally based on global properties of the surface. However, if an object-centered coordinate system can be extracted robustly from surface data, then is view independence prompts its use over viewer-centered coordinate system.

In 3-D object recognition, an important application of surface matching, an object surface is searched for in a scene surface. Real scenes contain multiple objects, so surface data sensed in the real world will contain clutter, surfaces that are not part of the object surface being matched. Because clutter will corrupt global properties of the scene data, generating object-centered coordinate systems in the cluttered scenes is difficult. The usual method for dealing with clutter is to segment the scene into object and non-object components [1][8]; naturally, this is difficult if the position of the object is unknown. An alternative to segmentation is to construct object-centered coordinate systems using local features detected in the scene [10][20]; here again there is the problem of differentiating object features from non-object features. Another difficulty occurs because surface data often has missing components i.e., occlusions. Occlusions will alter global properties of the surfaces and therefore, will complicate construction of object-centered coordinate systems. Consequently, if an object centered surface matching representation is to be used to recognize objects in real scenes, it must be robust to clutter and occlusion.

Object representations should also enable efficient matching of surfaces from multiple models, so that recognition occurs in a timely fashion. Furthermore, the representation 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 in the real world.

1.1 A Representation for Surface Matching

In our representation, surface shape is described by a dense collection of 3-D points and surface normals. In addition, associated with each surface point is a descriptive image that encodes global properties of the surface using an object-centered coordinate system. By matching images, correspondences between surface points can be established and used to match surfaces independent of the transformation between surfaces. Taken together, the points, normals and associated images make up our surface representation. Figure 1 shows the components of our surface matching representation.

Representing surfaces using a dense collection of points is feasible because many 3-D sensors and sensing algorithms return a dense sampling of surface shape. Furthermore, from sensor geometry and scanning patterns, the adjacency on the surface of sensed 3-D points can be established. Using adjacency and position of sensed 3-D points surface normal can be computed. We use a polygonal surface mesh to combine information about the position of 3-D surface points and the adjacency of points on the surface. In a surface mesh, the vertices of the surface mesh correspond to 3-D surface points, and the edges between vertices convey adjacency. Given enough points, any object can be represented by points sensed on the object surface, so surface meshes can represent objects of general shape. Surface meshes can be generated from different types of sensors and do not generally contain sensor-specific information; they are sensor-independent representations. The use of surface mesh as representations for 3-D shapes has been avoided in the past due to computational concerns. However, our research and the findings of other researchers have shown that processing power has reached a level where computations using surface meshes are now feasible [2][28].

Our approach to surface matching is based on matching individual surface points in order to match complete surfaces. Two surfaces are said to be similar when the images from many points on the surfaces are similar. By matching points, we are breaking the problem of surface matching into many smaller localized problems. Consequently, matching points provides a method for handling clutter and occlusion in surface matching without first segmenting the scene; clutter points on one surface will not have matching points on the other, and occluded points on one surface will not be searched for on the other. If many points between the two surfaces match then the surfaces can be matched. The main difficulty with matching surfaces in this way is describing surface points so that they can be differentiated from one another, while still allowing point matching in scenes con-

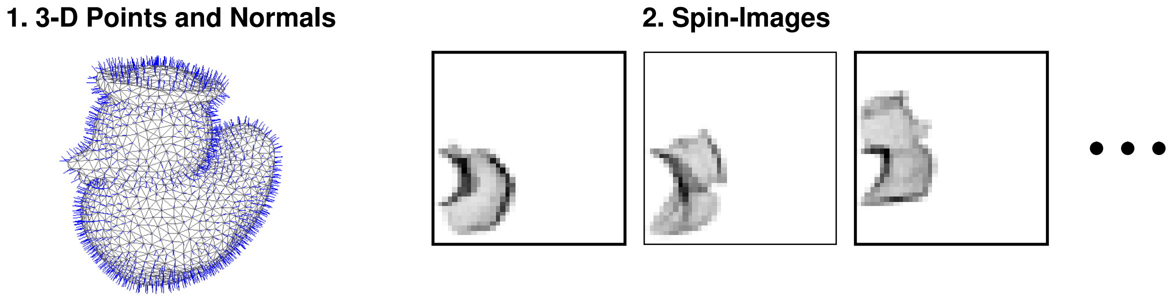


Figure 1: Components of our surface representation. A surface described by a polygonal surface mesh can be represented for matching as a set of 3-D points, surface normals and spin-images.

taining clutter and occlusion.

The idea of matching points to match surfaces is not a novel concept. Stein and Medioni [26] recognize 3-D objects by matching points using structural indexing and their “splash” representation. Similarly, Chua and Jarvis [3] match points to align surfaces using principal curvatures and “point-signatures”[4] Our methods differs from these in the way that points are represented for matching and the way that points, once matched, are grouped to match surfaces. Instead of matching points using 3-D curves that need to be aligned before matching, we match points using a rotationally invariant image. Because we use and image based representation, many of the techniques from image processing can be used in our matching algorithms. Consequently, our representation allows for more elegant solutions to point-based surface matching.

To differentiate among points, we construct 2-D images associated with each point. These images are created by constructing a local basis at an oriented point (3-D point with surface normal) on the surface of an object. As in geometric hashing [20], the positions with respect to the basis of other points on the surface of the object can then be described by two parameters. By accumulating these parameters in a 2-D histogram, a descriptive image associated with the oriented point is created. Because the image encodes the coordinates of points on the surface of an object with respect to the local basis, it is a local description of the global shape of the object and is invariant to rigid transformations. Since 3-D points are described by images, we can apply powerful techniques from 2-D template matching and pattern classification to the problem of surface matching.

To distinguish our point matching representation from camera images common in computer vision we have chosen the name *spin-image*; *image* because the representation is a 2-D array of values, and *spin* because the image generation process can be visualized as a sheet spinning about the normal of the point. Previous papers [14] introduced the concept of spin-images and showed how they can be used to match surfaces. Therefore, in Section 2, we briefly review the spin-image generation and its application to surface matching. This section also presents an analysis of the parameters used in spin-image generation showing the effect of different parameter values on the accuracy of spin-image matching.

The main contribution of this paper is the description and experimental analysis of the use of spin-images in efficient multi-model object recognition in scenes containing clutter and occlusion. Two major improvements to spin-image matching enable efficient object recognition. First, localization of spin-images by reducing spin-image generation parameters enables surface matching in scenes containing clutter and occlusion. Second, since the large number of spin-images comprising our surface representation are redundant, statistical eigen-analysis can be employed to reduce the dimensionality of the images and speed up spin-image matching. The techniques employed are similar to those used in appearance based recognition [21], and in particular, the combination of localized images and image compression is similar to the work in eigen-features [23] and parts-based appearance recognition [13]. Section 3 describes our algorithm for multi-model object recognition using spin-images, and Section 4 describes our experimental validation of recognition using spin-images on 100 complex scenes. A shorter description of this work has appeared as a conference paper [16].

2 Surface Matching

This section provides the necessary background for understanding spin-image generation and surface matching using spin-images. More complete description of spin-images and our surface matching algorithms are given in [14][17].

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.

A spin-image is created for an oriented point at a vertex in the surface mesh as follows. A 2-D ac-

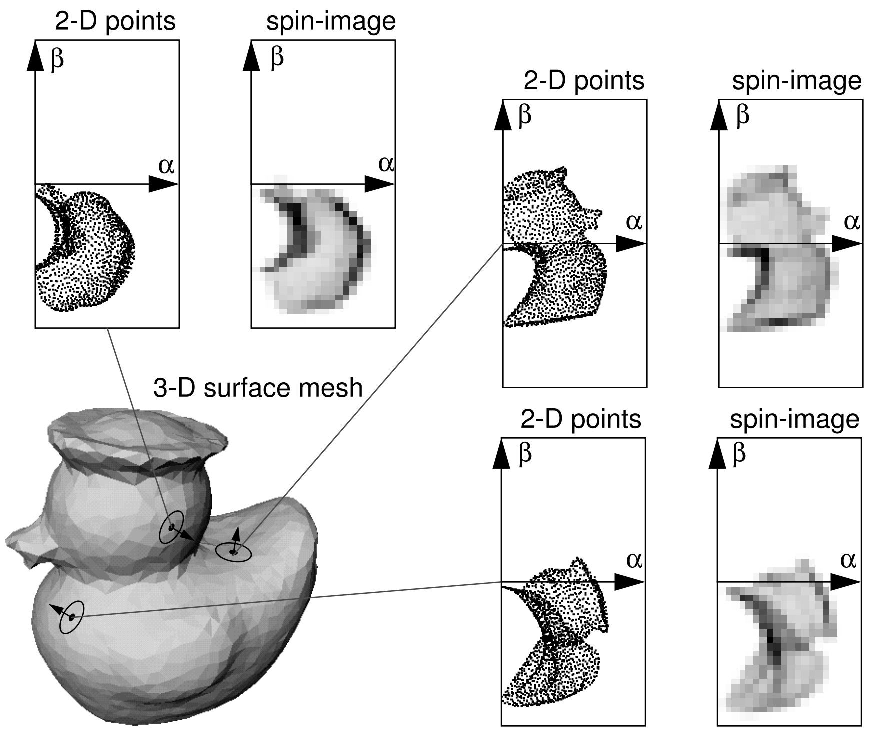


Figure 2: Spin-images of large support for three oriented points on the surface of a rubber duck model.

cumulator 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 greater than the median distance between vertices in the mesh (the definition of mesh resolution), the position of individual vertices will be averaged out during spin-image generation. Figure 2 shows the projected (α,β) 2-D coordinates and spin-images for three oriented points on a duck model. For surface matching, spin-images are constructed for every vertex in the surface mesh.

Spin-images generated from two different surfaces representing the same object will be similar because they are based on the shape of the object, but they will not be exactly the same due to variations in surface sampling and noise. However, if the surfaces are uniformly sampled then the spin-images from corresponding points on the different surfaces will be linearly related. (Uniform surface sampling is enforced by preprocessing the surface meshes using a mesh resampling algorithm [18].) A standard method for comparing linearly related data sets is the linear correlation coefficient, so we use correlation coefficient between two spin-images to measure spin-image similarity. As is shown in Section 3, for efficient object recognition, the similarity measure between spin-images must be changed to the distance between images. Distance between images performs as well as correlation coefficient for spin-image matching, as long as the images are properly normalized.

2.2 Spin-Image Generation Parameters

Bin size is the geometric width of the bins in the spin-image. Bin size is an important parameter in spin-image generation because it determines the storage size of the spin-image and the averaging in spin-images that reduces the effect of individual point positions. It also has an effect on the descriptiveness of the spin-images. The bin size is set as a multiple of the resolution of the surface mesh in order to eliminate the dependence of setting bin size on object scale and resolution. Setting bin size based on mesh resolution is feasible because mesh resolution is related to the size of shape features on an object and the density of points in the surface mesh. Spin-images generated for the duck model using different bin sizes are shown in Figure 3. The spin-image generated for a bin size of four times the model resolution is not very descriptive of the global shape of the model. The spin-image generated with a bin size of one quarter the mesh resolution does not have enough averaging to eliminate the effect of surface sampling. The spin-image generated with a bin size equal to the mesh resolution has the proper balance between encoding global shape and averaging of point positions

Figure 6 gives a quantitative analysis of the effect of bin size on spin-image matching. To create the graph, first, the spin-images for all vertices on the model were created for a particular bin size. Next, each spin-image was compared to all of the other spin-images from the model, and the Euclidean distances between the vertex and the vertices corresponding to the best matching spin-images were computed. After repeating this matching for all spin-images on the model, the median Euclidean distance (match distance) was computed. By repeating this procedure for multiple bin

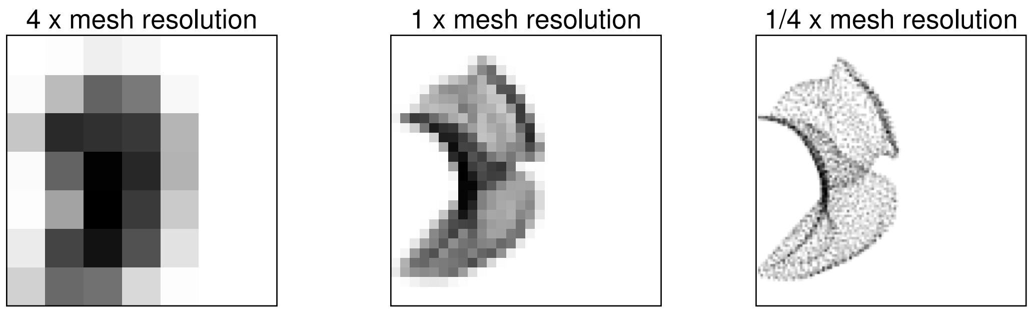


Figure 3: The effect of bin size on spin-image appearance. Three spin-images of decreasing bin-size for a point on the duck model are shown. Setting the bin size to the model resolution creates descriptive spin-images while averaging during point accumulation to eliminate the effect of individual vertex positions.

sizes using the duck model, that graph at the top of Figure 6 was created. Match distance is a single statistic that describes the correctness of spin-image matches, the lower the match distance, the more correct the matches.

The graph shows that for bin sizes below the mesh resolution (1.0) the match distance is large while for bin sizes greater than the mesh resolution, the match distance increases. Consequently, the best spin-image matching occurs when bin-size is set close to the mesh resolution; this analysis confirms our qualitative observations from Figure 3. For the results in this paper, the bin size is set to the exactly the mesh resolution.

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

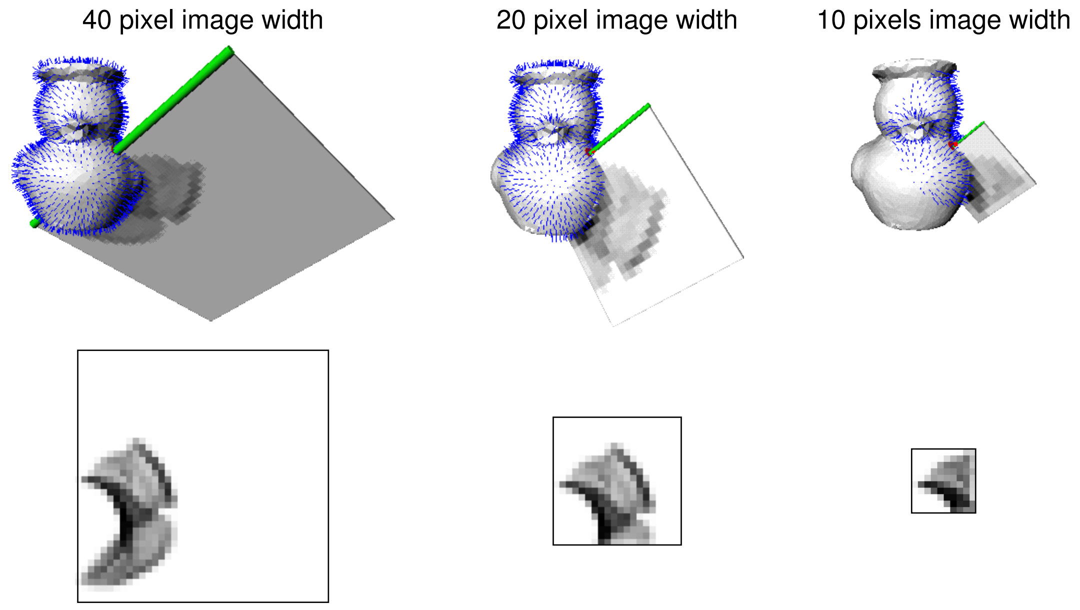


Figure 4: The effect of image width on spin-images. As image width decreases, the volume swept out by the spin-image(top) decreases, resulting in decreased spin-image support (bottom). By varying the image width, spin-images can vary smoothly from global to local representations.

port distance determines the amount of space swept out by a spin-image. By setting the image width, the amount of global information in a spin-image can be controlled. For a fixed bin-size, decreasing image width will decrease the descriptiveness of a spin-image because the amount of global shape included in the image will be reduced. However, decreasing image width will also reduce the chances of clutter corrupting a spin-image. Image width is analogous to window size in 2-D template matching. Figure 4 shows spin-images for a single oriented point on the duck model as the image width is decreased. This figure shows that as image width decreases, the descriptiveness of the images decreases.

Figure 6 shows the effect of image width on spin-image matching. As image width decreases, match distance decreases. This confirms our observation from Figure 4. In general, we set the image width so that the support distance is on order of the size of the model. If the data is very cluttered, then we set the image width to a smaller value. For the results presented in this paper, image width is set to 15, resulting in spin-images with 225 bins.

The final spin-image generation parameter is *support angle* A_s. Support angle is the maximum angle between the direction of the oriented point basis of a spin-image and the surface normal of

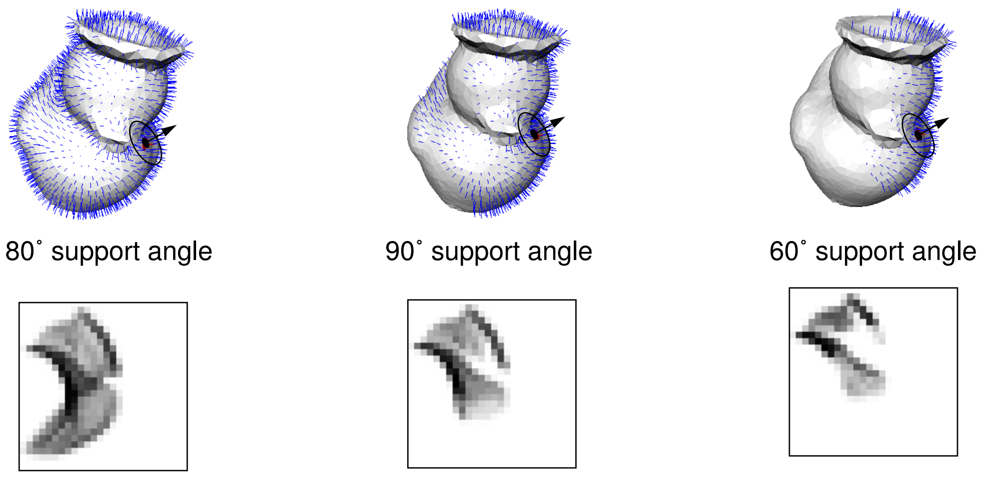


Figure 5: The effect of support angle on spin-image appearance. As support angle decreases, the number of points contributing to the spin-image (top) decreases. This results in reduction in the support of the spin-images (bottom).

points that are allowed to contribute to the spin-image. Suppose we have an oriented point $\mathcal{A}$ with position and normal $(\boldsymbol{p}_{\mathcal{A}},\boldsymbol{n}_{\mathcal{A}})$ for which we are creating a spin-image. Furthermore, suppose there exists another oriented point $\mathcal{B}$ with position and normal $(\boldsymbol{p}_{\mathcal{B}},\boldsymbol{n}_{\mathcal{B}})$. The support angle constraint can then be stated as: $\mathcal{B}$ will be accumulated in the spin-image of $\mathcal{A}$ if

$$\operatorname{acos}(\boldsymbol{n}_{\mathcal{A}} \cdot \boldsymbol{n}_{\mathcal{B}}) < A_s \quad . \tag{5.1}$$

Support angle is used to limit the effect of self occlusion and clutter during spin-image matching;. Figure 5 shows the spin-image generated for three different support angles along with the vertices on the model that are mapped into the spin-image. Support angle is used to reduce the number of points on the opposite side of the model that contribute to the model spin-image. This parameter decreases the effect of occlusion on spin-image matching; if a point has significantly different normal from the normal of the oriented point, then it is unlikely that it will be visible when the oriented point is imaged by a rangefinder in some scene data.

Decreasing support angle also has the effect of decreasing the descriptiveness of spin-images. Figure 6 shows the effect of support angle on spin-image match distance. The graph shows that as support angle decreases, the match distance increases because the spin-images are becoming less descriptive. However, a small support angle is necessary for robustness to clutter and occlusion.

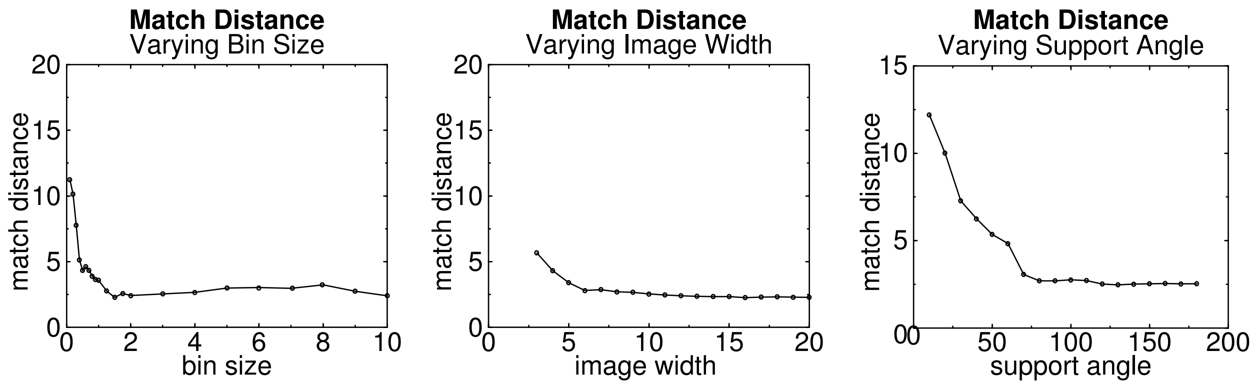


Figure 6: Effect of spin-image generation parameters bin size, image width and support angle on match distance.

We found that a balance can be struck between shape descriptiveness and matching robustness; in this paper all results are generated for a spin-images generated with a support angle of 60°.

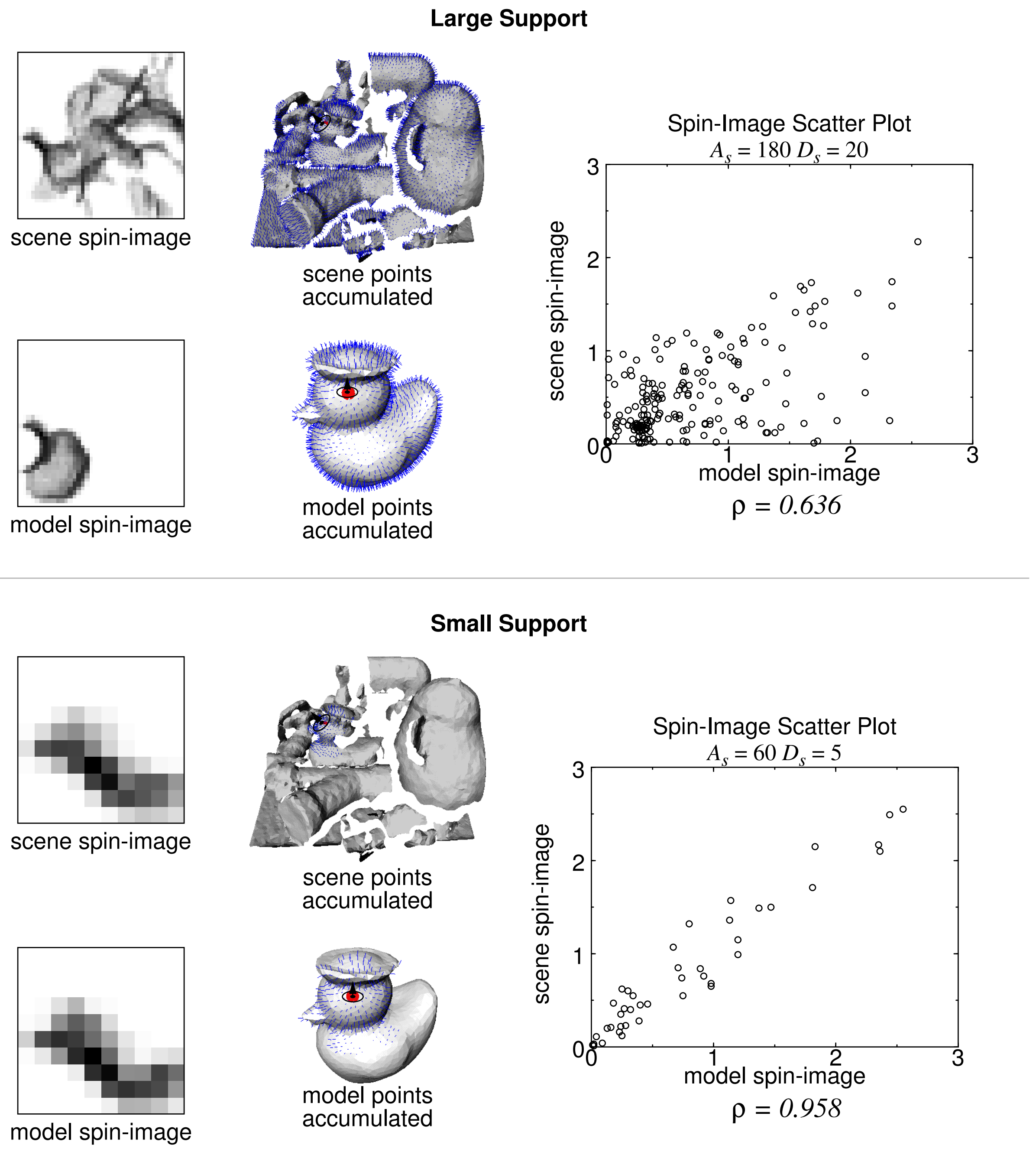


Figure 7: Localizing generation parameters increases the similarity of spin-images. The top shows a scatter plot of the model and scene spin-images generated using global parameters. The scatter plot shows that the spin-images are not particularly correlated. The bottom shows a scatter plot of the model and scene spin-images generated using local parameters. The scatter plot shows that the spin-images are much more linearly correlated. Localizing the spin-images throws away image pixels where the images disagree.

Figure 7 shows how spin-image generation parameters localize spin-images to reduce the effect of scene clutter on matching. When the spin-images are not localized, the model and scene spin-images are very different in appearance, a fact which is born out by the correlation coefficient of the two images (ρ = 0.636). As the spin-images are localized by decreasing the support angle A_S and support distance D_S, the spin-images become much more similar. When creating spin-images with large support angle and distance, many scene points that do not belong to the model are spin-mapped into the scene spin-image. This causes the scene spin-image to become uncorrelated with the model spin-image because, as shown in the scatter plot of the two images, scene spin-image pixels are being corrupted by clutter. When smaller support angle and distance are used, the spin-images become similar; the pixel values shown in the scatter plot of the images created with local parameters are linearly related. (ρ = 0.958)By varying spin-image generation parameters, we are using knowledge of the spin-image generation process to eliminate outlier pixels, making the spin-images much more similar.

2.3 Surface matching engine

As shown in Figure 8, two surfaces are matched as follows. Spin-images from points on one surface are compared by computing 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. More specifically, before matching, all of the spin-images from one surface (the model) are constructed and stored in a spin-image stack. Next, a vertex is selected at random from the other surface (the scene) and its spin-image is computed. Point correspondences are then established between the selected point and the points with best matching spin-images on the other surface. This procedure is repeated for many points resulting in a sizeable set of point correspondences (~100). Point correspondences are then grouped and outliers are eliminated using geometric consistency. Groups of geometrically consistent correspondences are then used to calculate rigid transformations that aligns one surface with the other. After alignment, surface matches are verified using a modified iterative closest point algorithm. The best match is selected as the one with the greatest overlap between surfaces. Further details of the surface matching engine are given in [14].

3 Object recognition

Surface matching using spin-images can be extended to object recognition as follows. Each model

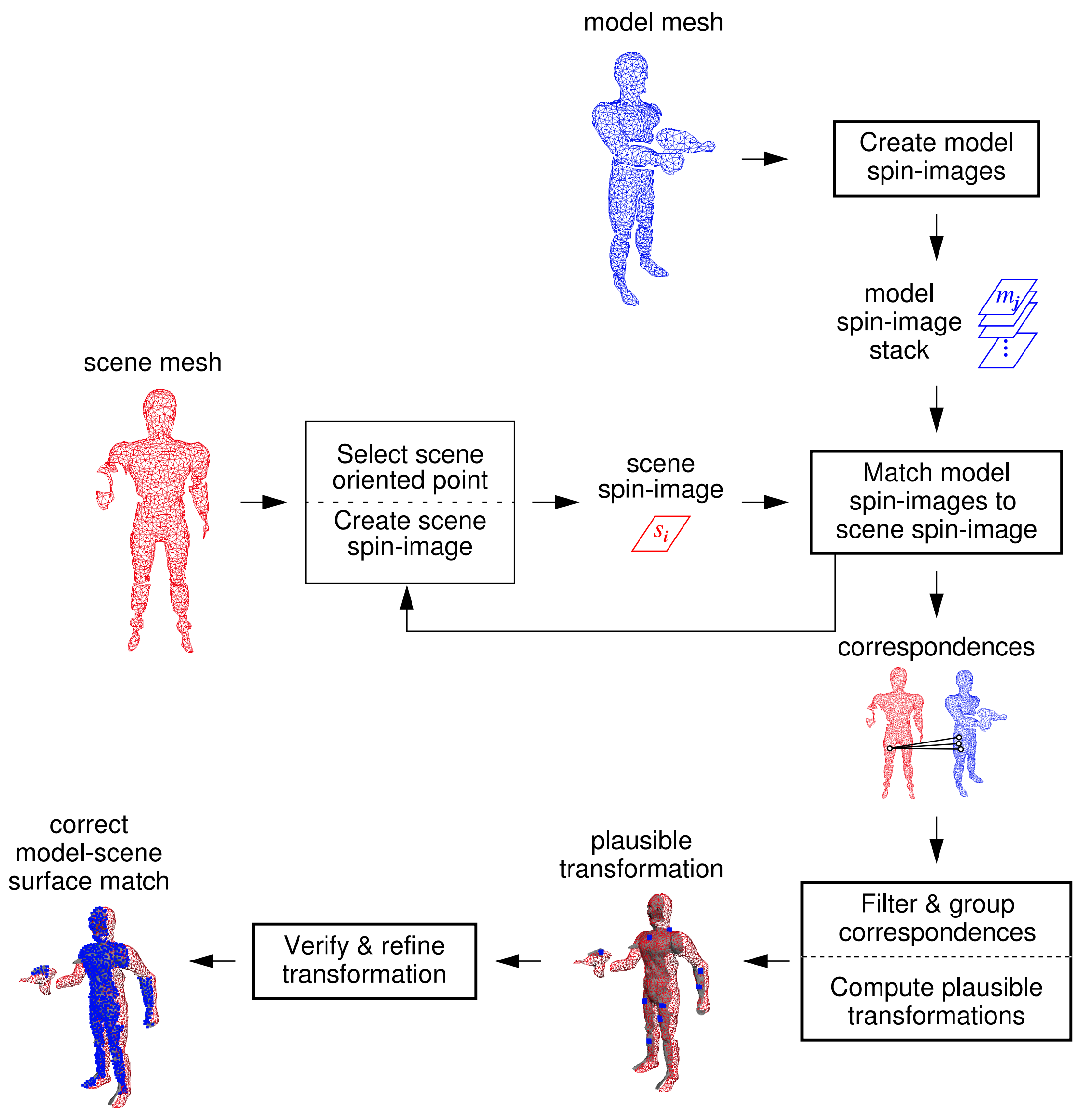

model mesh
Create model spin-images
model spin-image stack
m_j
scene mesh
Select scene oriented point
Create scene spin-image
scene spin-image
s_i
Match model spin-images to scene spin-image
correspondences
correct model-scene surface match
plausible transformation
Filter & group correspondences
Compute plausible transformations
Verify & refine transformation

Figure 8: Surface matching block diagram.

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. Next, its 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 2.3. The result is simultaneous recognition and localization of the models that exist in the scene.

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 speed up matching considerably.

3.1 Spin-Image Compression

Spin-images coming from the same surface can be correlated for two reasons: First, as shown in Figure 9, spin-images generated from oriented point bases that are close to each other on the surface will be correlated. Second, as shown in Figure 9, surface symmetry and the inherit symmetry

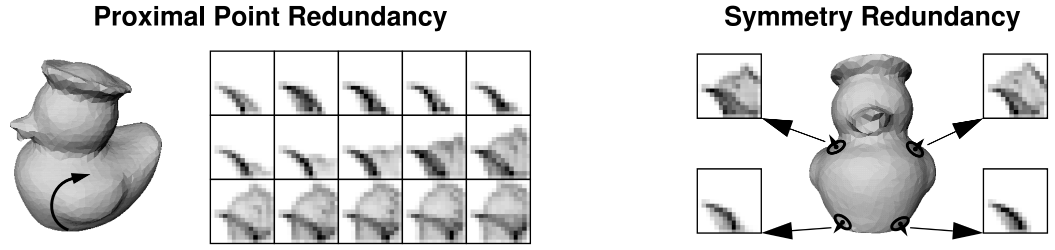


Figure 9: Spin-images generated while traversing a path along the surface of the duck model (left). Spin-images from proximal oriented points are similar, resulting in one cause of redundancy in spin-images.Two pairs of similar spin-images caused by symmetry in the duck model (right).

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, so 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 an 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)[21]. PCA or Karhunen-Loeve expansion is a well known method for computing the directions of greatest variance for a set of vectors [9]. 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. The l_2 distance between two spin-images in spin-image space is the same as the l_2 distance between the two spin-images represented in the eigenspace. Furthermore, when vectors are projected into a subspace defined by the eigenvectors of largest eigenvalue, the l_2 distance between projected vectors is the best approximation (with respect to mean square error) to the l_2 distance between the unprojected vectors, given the dimension of the subspace [9]. By minimizing mean-square error, PCA gives us an elegant way to balance compression of images against ability to discriminate between images.

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; the mean of all of the spin-images in the library is

$$\bar{\boldsymbol{x}} = \sum_{i=1}^{N} \boldsymbol{x}_i . \qquad (1)$$

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

$$\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i - \bar{\boldsymbol{x}} \qquad (2)$$

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

$$S^m = \left[\hat{\boldsymbol{x}}_1 \ \hat{\boldsymbol{x}}_2 \ \dots \ \hat{\boldsymbol{x}}_N\right] . \qquad (3)$$

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

$$C^m = S^m (S^m)^T . \qquad (4)$$

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

$$\lambda_i^m \boldsymbol{e}_i^m = C^m \boldsymbol{e}_i^m . \qquad (5)$$

Since the dimension of the spin-images is not too large (~200), the standard **`jacobi`** algorithm from Numerical Recipes in C [24] is used to determine the eigenvectors $\boldsymbol{e}_j^m$ and eigenvalues λ_j^m of C^m. Since the eigenvectors of C^m can be considered spin-images, they will be 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[17]). Every spin-image from each model is then projected into the s-dimensional subspace spanned by the s eigenvectors of largest eigenvalue; the s-tuple of projection coefficients, $\boldsymbol{p_j}$, becomes the compressed representation of the spin-image.

$$p_j = (\hat{x}_j e_1^m, \hat{x}_j e_2^m, \ldots, \hat{x}_j e_s^m) \tag{6}$$

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 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 for the scene spin-image. The scene spin-image is generated using the model spin-image generation parameters. Suppose the scene spin-image for scene oriented point *j* is represented in vector notation as y_j. The first step in constructing the scene *s*-tuple is to subtract the mean of the model spin-images

$$\hat{y}_j = y_j - \bar{x} \tag{7}$$

Next the mean-subtracted scene spin-image is projected onto the top *s* library eigen-spin-images to get the scene *s*-tuple q_j

$$q_j = (\hat{y}_j e_1^m, \hat{y}_j e_2^m, \ldots, \hat{y}_j e_s^m) \tag{8}$$

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

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

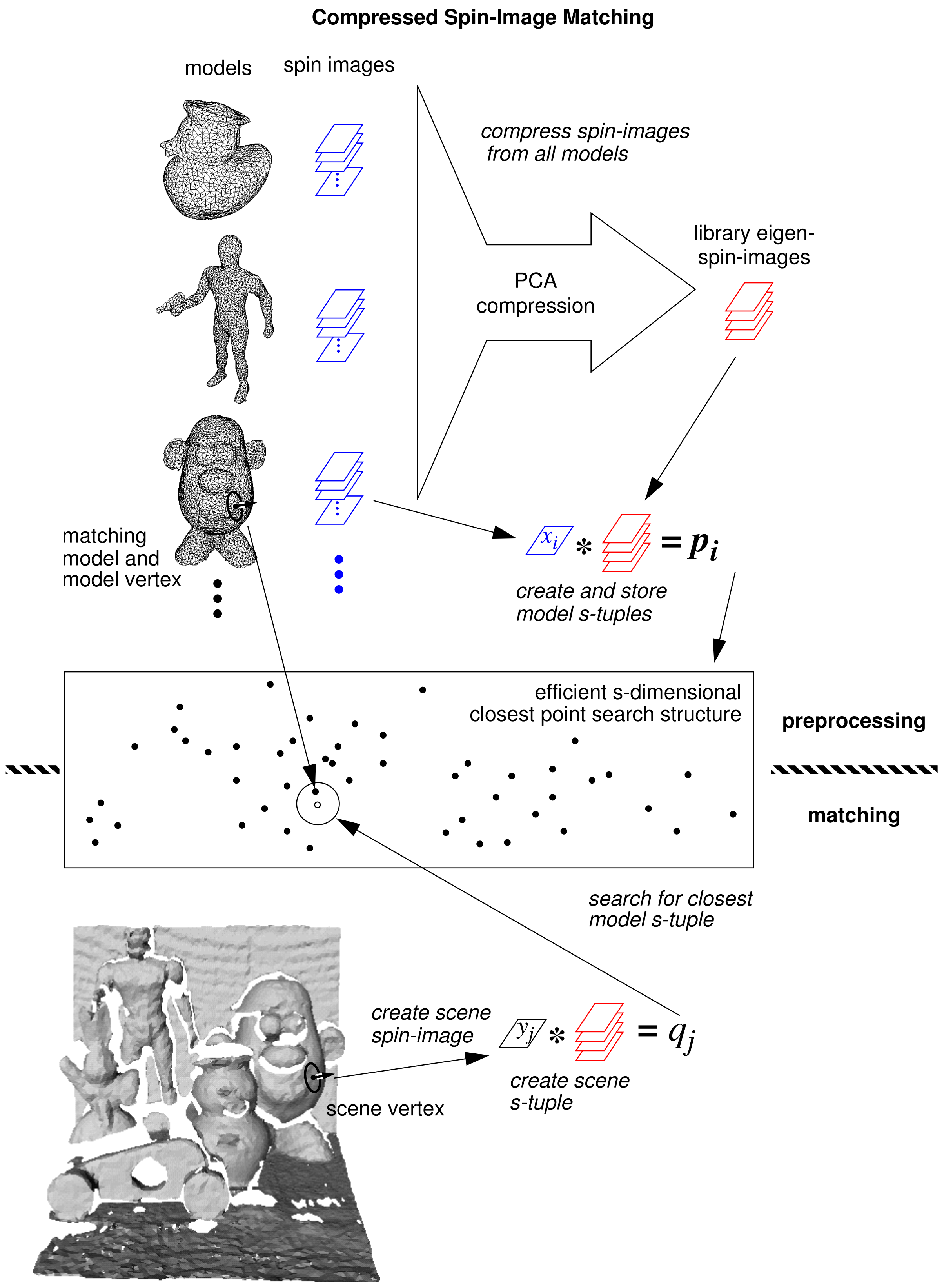

Compressed Spin-Image Matching
models
spin images
compress spin-images from all models
PCA compression
library eigen-spin-images
matching model and model vertex
x_i * $= p_i$
create and store model s-tuples
efficient s-dimensional closest point search structure
preprocessing
matching
search for closest model s-tuple
create scene spin-image
y_j * $= q_j$
create scene s-tuple
scene vertex

Figure 10: Procedure for simultaneous matching of multiple models to a single scene point.

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 [22]. 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 [21], a domain that is similar to spin-image matching. In both domains, PCA is used to compress images resulting in set of structured s-tuples that must be searched for closest points. In out implementation, the search parameter ε was automatically set to the median of the distances between pairs of closest 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. Figure 10 shows a pictorial description for the procedure for matching of multiple models to a single scene point. 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 then stored in an 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, its spin-image is computed using the scene data. Next, for each model, the scene spin-image is projected onto the model's eigen-spin-images to obtain a scene s-tuple. The scene s-tuple is then used as a query point into the current model's efficient closest point search structure which returns a list of current model s-tuples close to the scene s-tuple. These point matches 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 11; each was created by registering and integrating multiple range views of the objects [15]. 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 [27] and mesh resampling [18] to change the scene data resolution to that of the models in the model library. In all of the following results, the spin-image generation parameters are: a bin-size equal to mesh

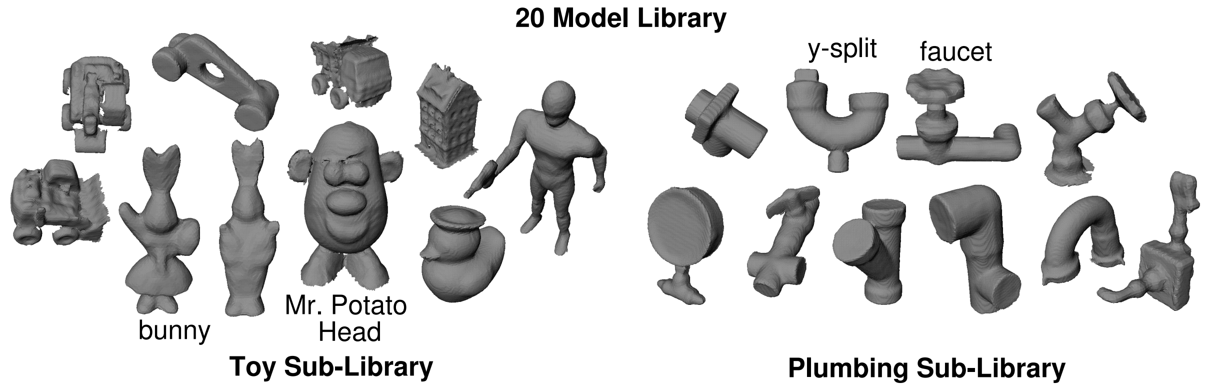


Figure 11: 20 Models used for recognition.

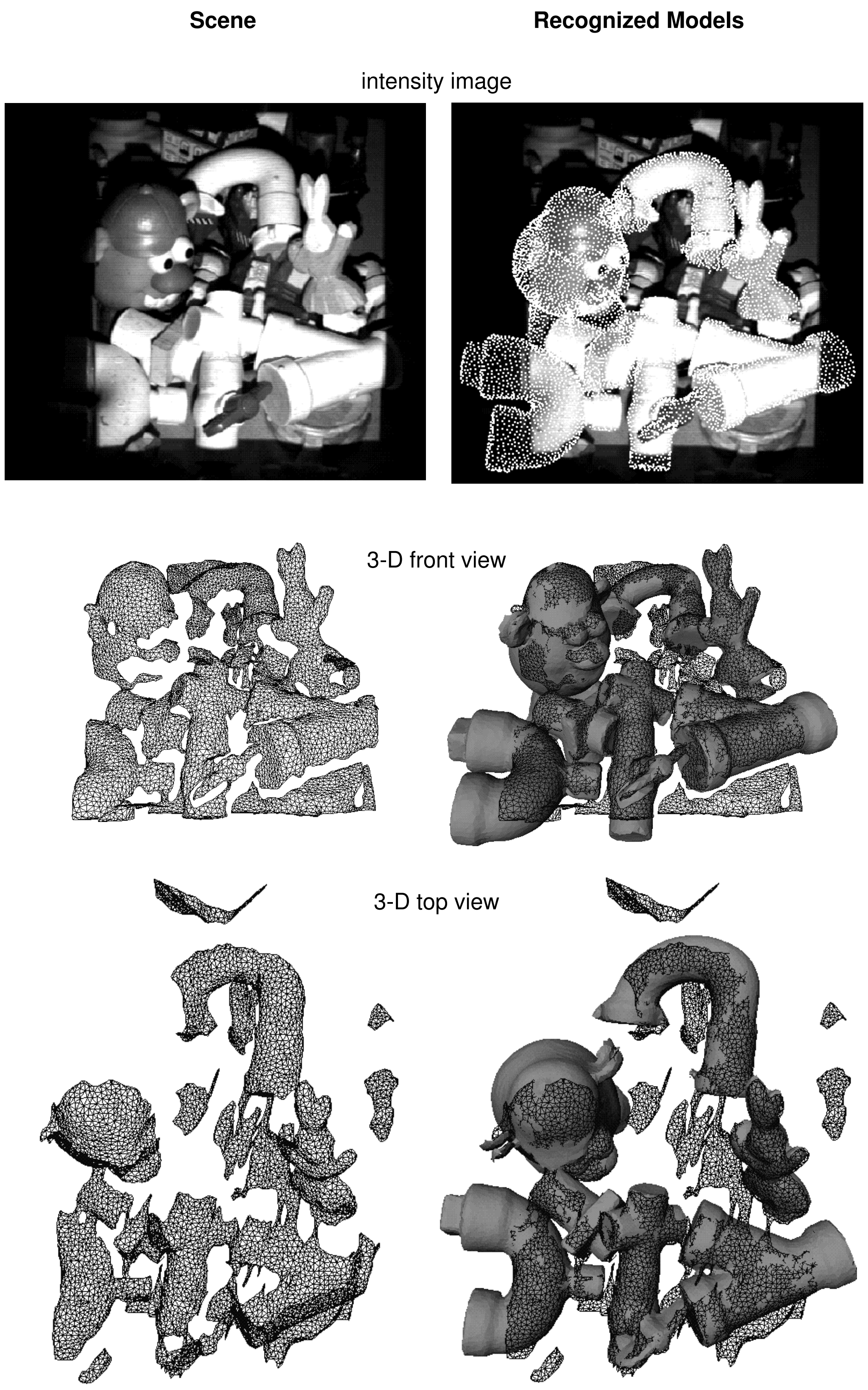


Figure 12: Simultaneous recognition of 7 models from a library of 20 models in a cluttered scene.

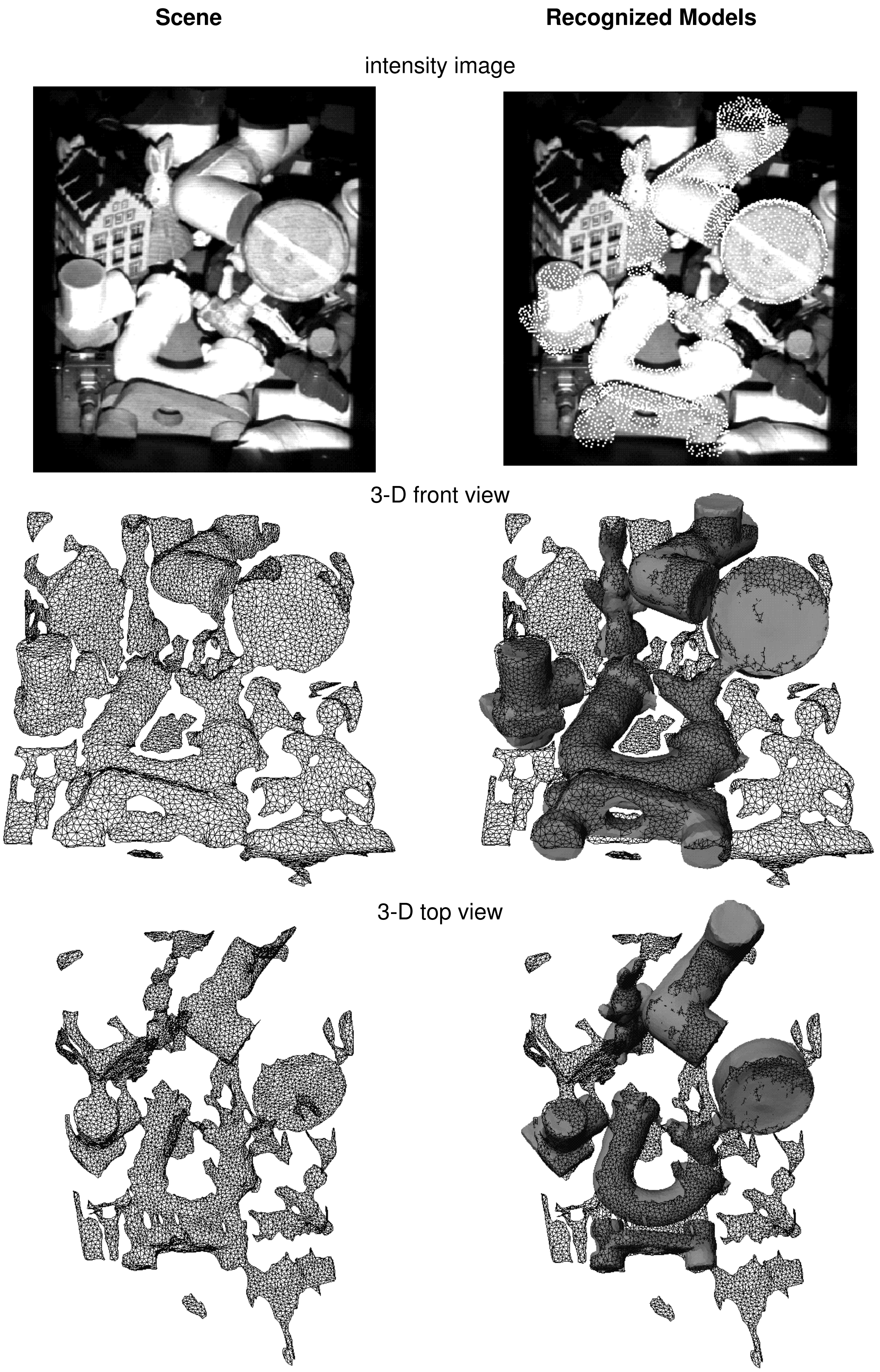


Figure 13: Simultaneous recognition of 6 models from a library of 20 models in a cluttered scene.

resolution, an image width of 15 bins (225 bins per image) and a support angle of 60˚.

Figure 12 shows the simultaneous recognition of seven models from the library of twenty models. 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

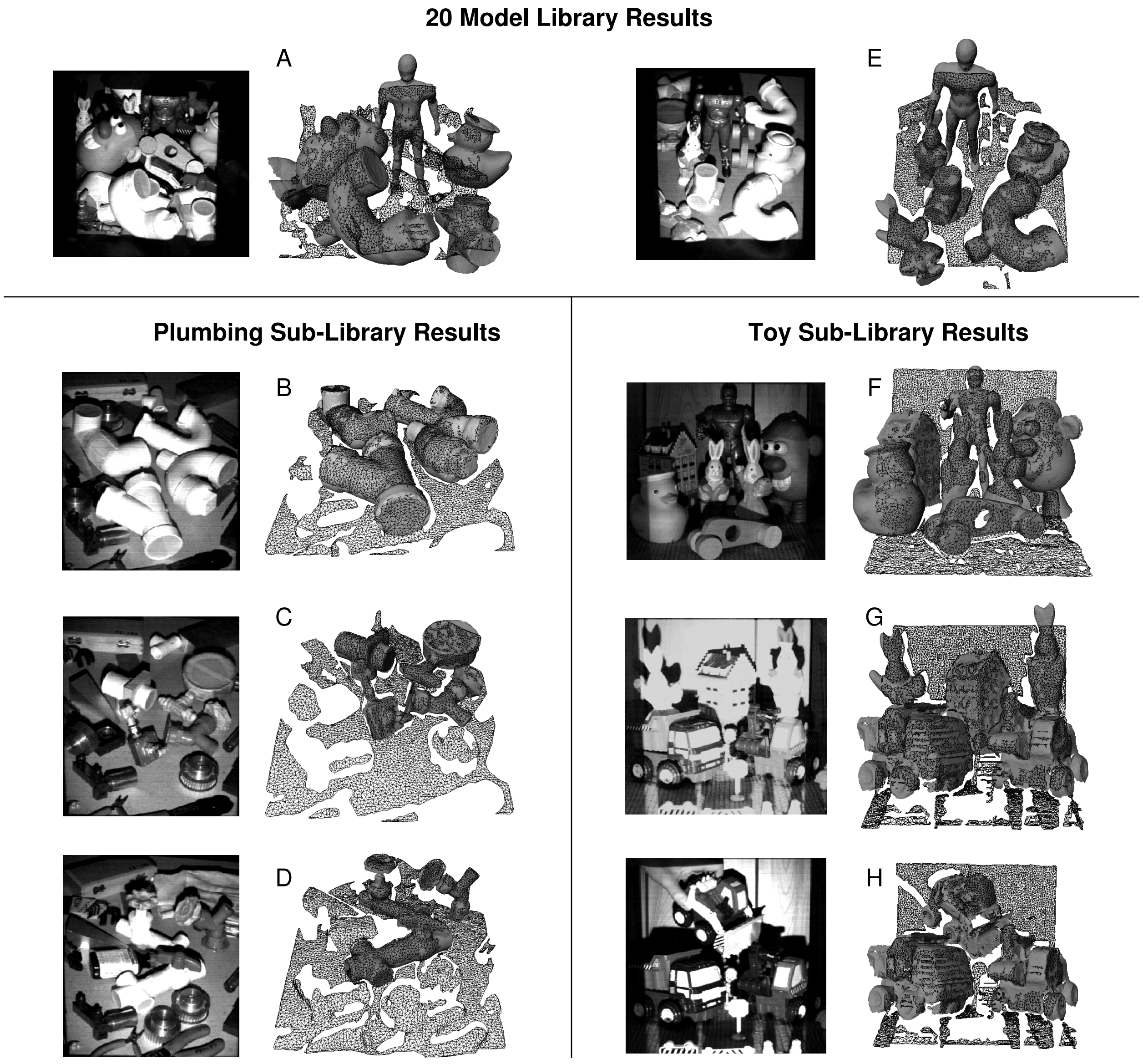


Figure 14: Additional recognition results using the 20 model library, plumbing library and toy library shown in Figure 11. Each result shows a scene intensity image and a recognition result with recognized models overlaid on the scene surface mesh

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. Because spin-image matching has been designed to be resistant to clutter and occlusion, our algorithm is able to recognize simultaneously 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 13 shows the simultaneous recognition of 6 objects from a library of 20 objects in a similar format to Figure 12. Figure 14 shows some additional results using the different libraries shown in Figure 11. These results show that objects can be distinguished even when multiple object of similar shape appear in the scene (results B,D,F,G,H). They also show that recognition does not fail when a significant portion of the scene surface comes from objects not in the model library (results B,C,D).

4 Analysis of Recognition in Complex Scenes

Any recognition algorithm designed for the real world must work in the presence of clutter and occlusion. In Section 2, 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.

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.

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

$$\text{occlusion} = 1 - \frac{\text{model surface patch area}}{\text{total model surface area}} \tag{9}$$

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

$$\text{clutter} = \frac{\text{clutter points in relevant volume}}{\text{total points in relevant volume}} \tag{10}$$

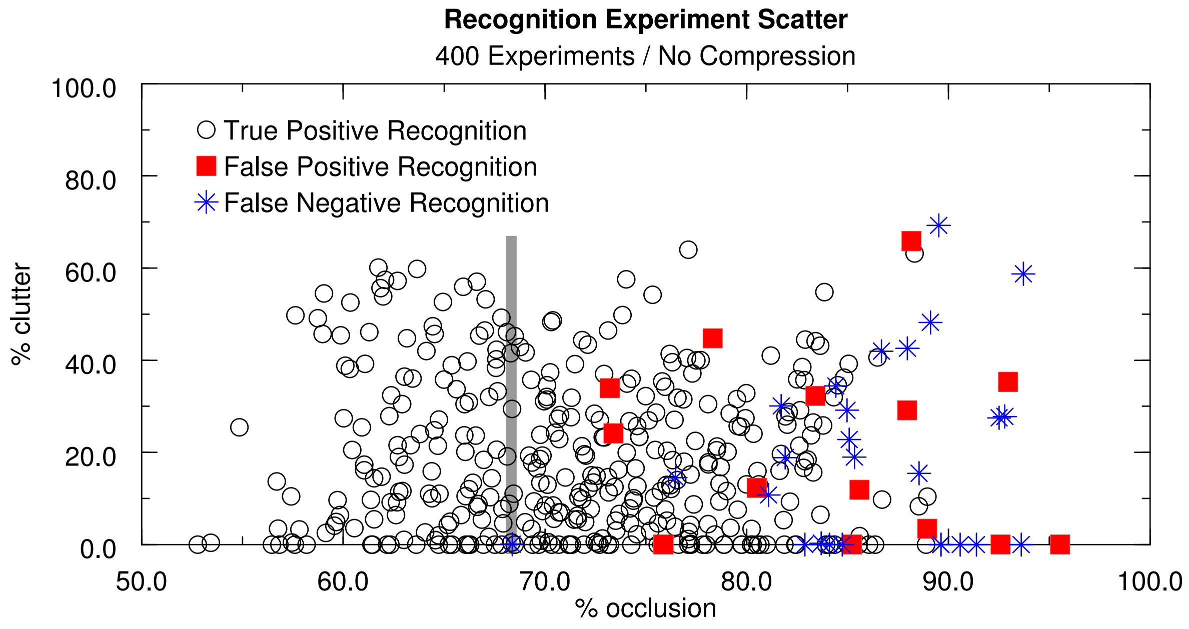
Clutter points are vertices in the scene surface mesh that are not on the model surface patch. The relevant volume is the union of the volumes swept out by each spin-image of all of the oriented points on the model surface patch. If the relevant volume contains points that are not on the model surface 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 11). 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. Using four models, we hoped to adequately sample the possible shapes to be recognized, given that sampling of all possible surface shapes is not experimentally feasible.

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 of Figure 15. 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 of Figure 15. Briefly looking at both scatter plots shows that the number of true-positive states is much larger than the number of false negative states and false-positive state. 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 15, one notices that recognition rate is effected by occlusion.



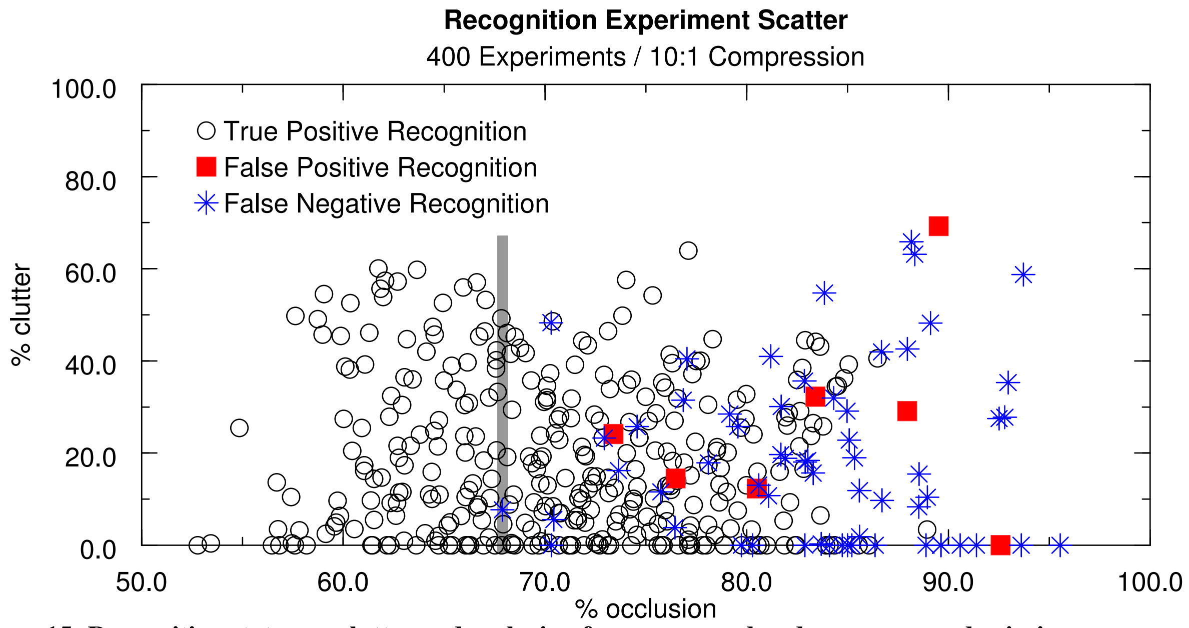


Figure 15: Recognition states vs. clutter and occlusion for compressed and uncompressed spin-images.

At low occlusion values, no recognition failures are reported, while at high occlusion values, recognition failures dominate. This indicates that recognition will almost always work if sufficient model surface area is visible. The decrease in recognition success after a fixed level of occlusion is reached (70%) indicates that spin-image matching does not work well when only a small portion of the model is visible. This is no surprise since spin-image descriptiveness comes from accumulation of surface area around a point. On the left in Figure 16 are shown the experimental recognition rates versus scene occlusion. The rates are computed using a gaussian weighted running average (averaging on occlusion independent of clutter level) 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 15, 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 16, 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 estimates of recognition rate are imprecise.

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 remain 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 rate for both algorithms are low and nearly the same.

The right graph in Figure 17 shows the result of an experiment that measured the average number of true positive recognitions for ten scenes versus the number of models in the model library. As the number of models in the library increases, the number of models correctly recognized increases linearly. This is caused by the model library containing more and more of the models that are present in the scene. The graph shows that matching without compression matches slightly more

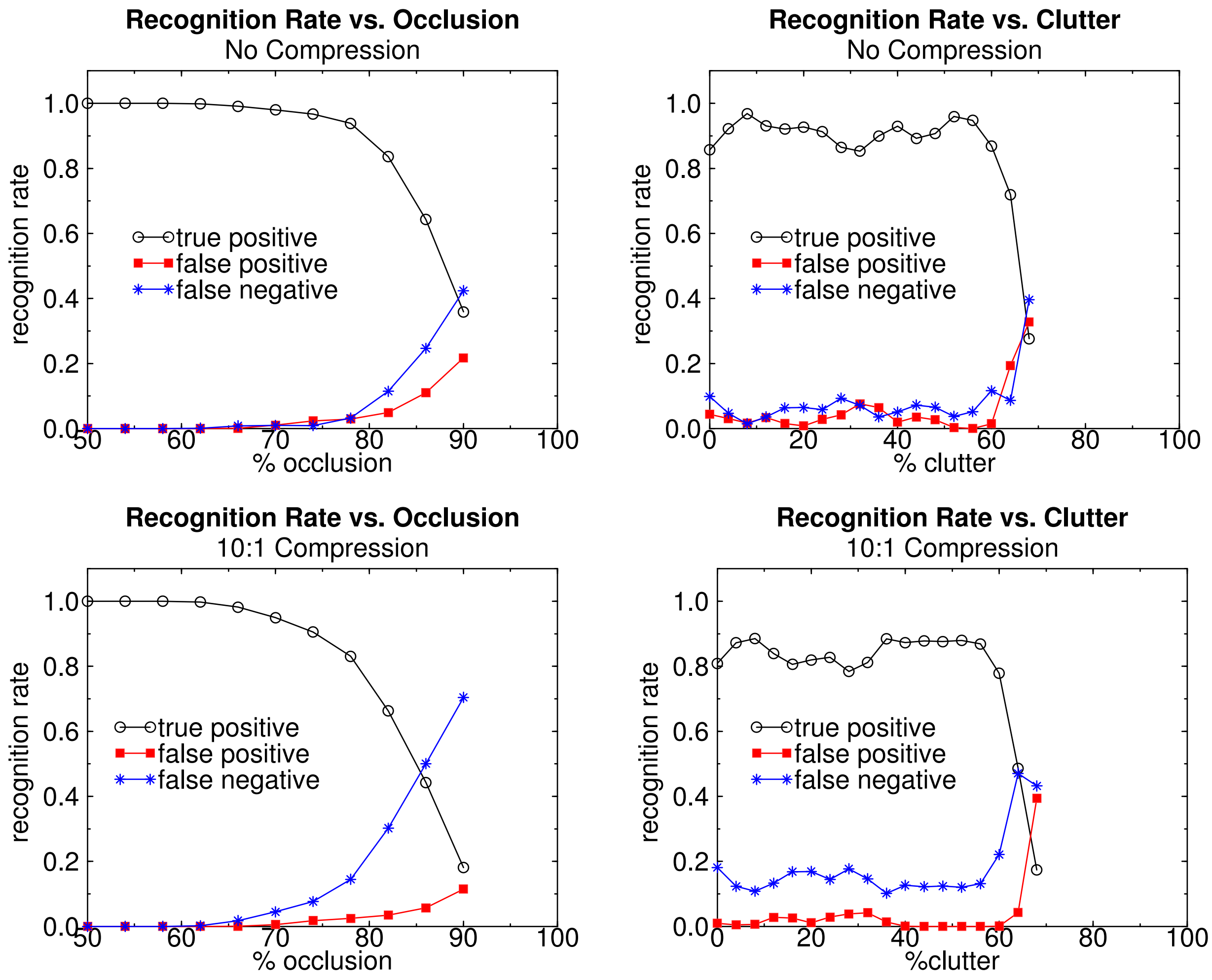
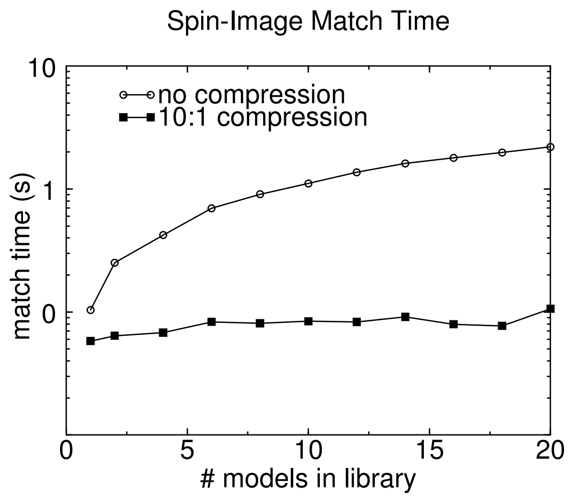


Figure 16: Recognition state probability vs. occlusion for compressed and uncompressed spin-images (middle). Recognition state probability vs. clutter (right).

models than matching with 10:1 compression, a consequence of uncompressed spin-images being more discriminating.

The time needed to match a single scene spin-image to all of the spin-images in the model library as the number of models in the library increases is shown in the graph on the left in Figure 17. All times are real wall clock times on a Silicon Graphics O2 with a 174 MHz R10000 processor. As expected, matching of spin-images grows linearly with the number of models in the model library because the number of spin-images being compared increases linearly with the number of models. This is true of matching with compression and matching without compression; however, the matching times with compression grow significantly slower than the matching times without compression. With twenty models in the library, matching with 10:1 compression is 20 times faster than matching without compression. Since there is only a slight decrease in recognition performance when using compression (right in Figure 17,) compressed spin-images should be used in recognition. Another factor in matching time, is the number of points in the scene. To obtain the total match time for the algorithms, the match times shown in Figure 17 should be multiplied by



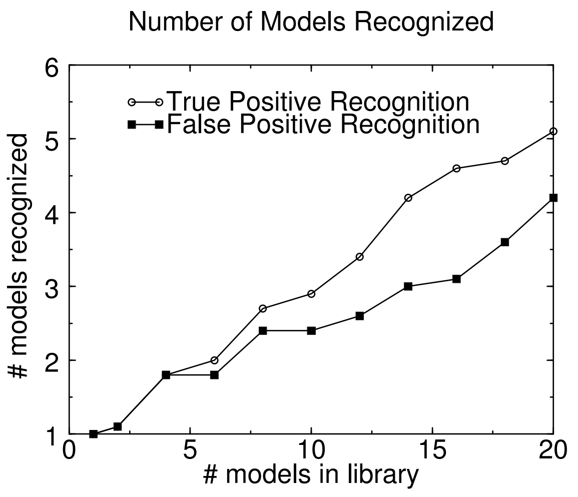


Figure 17: Numbers of models recognized (right) and spin-image matching time (left) vs. library size for compressed and uncompressed spin-images.

the number of points selected from the scene for matching.

5 Conclusion

We have presented an algorithm for simultaneous shape-based recognition of multiple objects in cluttered scenes with occlusion. Our algorithm can handle objects of general shape because it is based on the spin-image, 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. Finally 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.

Spin-images are a general shape representation, so their applicability to problems in 3-D computer vision is broad. This paper has investigated the application of spin-images to object recognition, however, other spin-image applications exist. For instance, the general nature of spin-images makes them an appropriate representation for shape analysis, the process that quantifies similarities and differences between the shape of objects. Shape analysis can lead to object classification, analysis of object symmetry and parts decomposition, all of which can be used to make object recognition more efficient.Other possible applications of spin-images include 3-D object tracking and volumetric image registration.

There still exist some algorithmic additions which could be implemented to make spin-image matching more efficient and robust. Some extensions currently being investigated are multi-resolution spin-images for coarse to fine recognition, automated learning of descriptive spin-images, and improved spin-image parameterizations.

Acknowledgments

We would like to thank Jim Osborn and all the members of the Artisan project for supporting this work. We would also like to thank Karun Shimoga for the use of the K^2T sensor, and Kaushik Merchant for his time spent segmenting 3-D scenes.

References

[1] F. Arman and J. K. Aggarwal. "CAD-based vision: object recognition in cluttered range images using recognition strategies." *Computer Vision, Graphics and Image Processing*, vol. 58, no. 1, pp. 33-48, 1993.

[2] P. Besl. "The triangle as primary representation." *Object Representation in Computer Vision*, M. Hebert, J. Ponce, T. Boult and A. Gross (Eds.), Springer Verlag, Berlin, 1995.

[3] 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.

[4] C. Chua and R. Jarvis. "Point Signatures: a new representation for 3D object recognition." *Int'l Jour. Computer Vision*, vol. 25, no. 1, pp. 63-85, 1996.

[5] 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.

[6] D. Dion Jr., D. Laurendeau and R. Bergevin. "Generalized cylinder extraction in range images." *Proc. Int'l Conf. on Recent Advance in 3-D Digital Imaging and Modeling,* Ottawa, pp. 141-147, May 1997.

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

[8] 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.

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

[10] W.E.L. Grimson. "Localizing overlapping parts by searching the interpretation tree." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 9, no. 4, pp 469-482, 1987.

[11] M. Hebert, K. Ikeuchi and H. Delingette. "A spherical representation for recognition of free-form surfaces." *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 17, no. 7, pp. 681-689, 1995.

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

[13] C-Y. Huang, O. Camps and T. Kanungo. “Object recognition using appearance-based parts and relations.” *Proc. Computer Vision and Pattern Recognition (CVPR ‘97),* pp. 877-883, May 1997.

[14] A. Johnson and M. Hebert. “Object recognition by matching oriented points.” *Proc. Computer Vision and Pattern Recognition (CVPR ‘97),* pp. 684-689, June1997.

[15] A. Johnson and S. Kang. “Registration and integration of textured 3-D data.” *Proc. Int’l Conf. on Recent Advance in 3-D Digital Imaging and Modeling (3DIM’97),* pp. 234-241, 1997.

[16] A. Johnson and M. Hebert. “Efficient multiple model recognition in cluttered 3-D scenes.” *Proc. Computer Vision and Pattern Recognition (CVPR ‘98),* pp. 671-678, 1998.

[17] A. Johnson. *Spin-Images: A representation for 3-D surface matching.* Ph.D Thesis, The Robotics Institute, Carnegie Mellon University, 1997.

[18] A. Johnson and M. Hebert. “Control of polygonal mesh resolution for 3-D computer vision.”. *Graphical Models and Image Processing*, too appear 1998.

[19] S. Kang and K. Ikeuchi. “The complex EGI: new representation for 3-D pose determination.” *IEEE Pattern Analysis and Machine Intelligence*, vol. 15, no. 7, pp. 707-721, 1997.

[20] 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.

[21] 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.

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

[23] K. Ohba and K. Ikeuchi. “Detectability, uniqueness and reliability of eigen-windows for stable verification of partially occluded objects.” *Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 1043-1130-1048, 1997.

[24] 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.

[25] 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.

[26] 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.

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

[28] G. Taubin. “Discrete surface signal processing: the polygon as the surface element.” *Object Representation in Computer Vision*, M. Hebert, J. Ponce, T. Boult and A. Gross (Eds.), Springer Verlag, Berlin, 1995.

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