Efficient Multiple Model Recognition in Cluttered 3-D Scenes

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

Martial Hebert
The Robotics Institute
Carnegie Mellon University
hebert@ri.cmu.edu

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.*

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:

- represent general shapes
- robust to clutter and occlusion
- 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 [4], super-quadrics [12] and spherical representations [2]). 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 [9], COSMOS [3]) 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 can be split 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 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.

This paper is organized as follows. In Section 2 we review the spin-image representation and show how spin-images can be used to match surfaces of arbitrary shape. Section 3 explains how correlation between spin-images can be exploited to compress spin-images for efficient multiple model object recognition. In Section 4 we present an analysis of hundreds of recognition trials that experimentally validate that recognition with spin-images is robust to clutter and occlusion.

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; a more complete description of spin-images and our surface matching algorithms are given in [6][7].

2.1 Spin-images

Oriented points, 3-D points with associated directions, are used to create spin-images. We define an oriented point

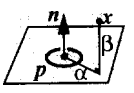

Figure 1. An oriented point basis defined at a surface mesh vertex.

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

0-8186-8497-6/98 $10.00

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 accumulator indexed by a 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 [7], 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.

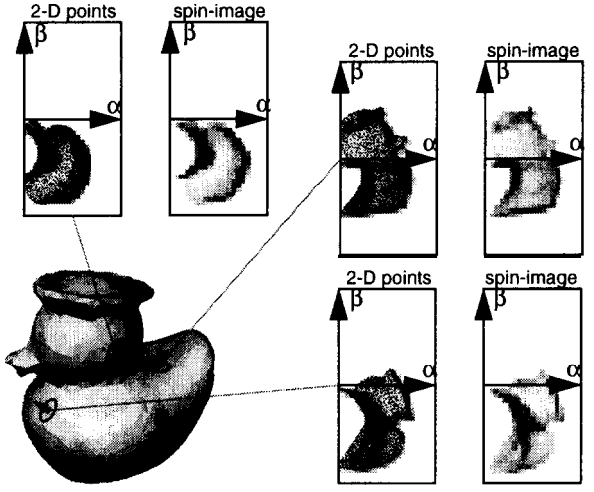

Figure 2. Spin-images of large support for three oriented points on the surface of a rubber duckie model.

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 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

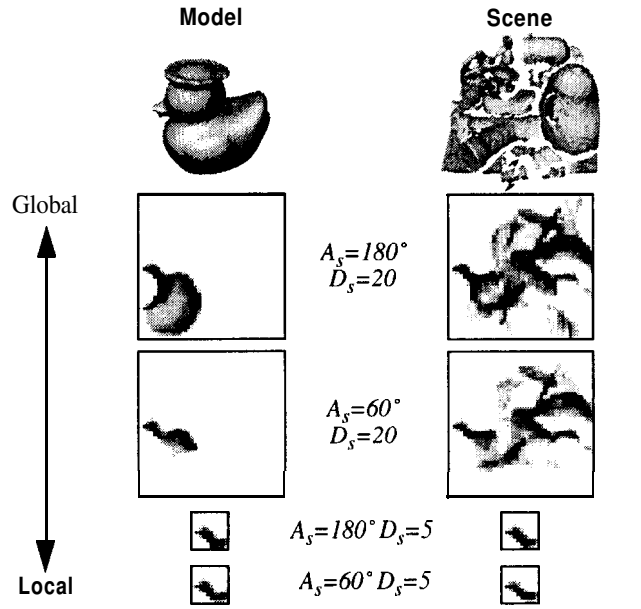

Figure 3. How spin-image generation parameters localize spin-images to reduce the effect of scene clutter and occlusion on matching.

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 [6].

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 there descriptiveness even in the presence of large sensor noise, clutter, occlusion, and local object symmetry[7].

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 its spin-image is correlated with all of the spin-images from ail 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.

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, 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 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)[9]. PCA or Karhunen-Loeve expansion [5] 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 [5].

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 $\boldsymbol{D}$. 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.

$$\dot{x}_i = x_i - \bar{x} \tag{1}$$

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

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

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

$$C^m = S^m (S^m)^T. \tag{3}$$

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

$$\lambda_i^m e_i^m = C^m e_i^m. \tag{4}$$

Since the dimension of the spin-images is not too large (~200), the standard `jacobi` algorithm from Numerical Recipes in C [11] 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 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 [7]). 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, p_j, becomes the compressed representation of the spin-image.

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

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 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-tupies. 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[10]. 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 [9], 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 [7] 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. 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

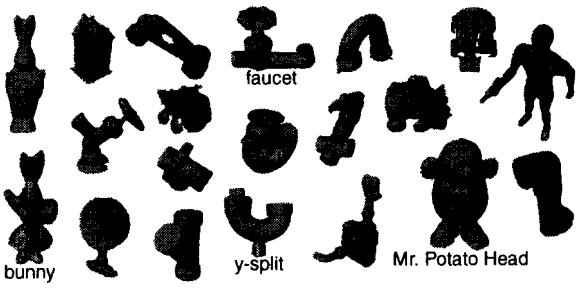

Figure 4. 20 Models used for recognition.

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.

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 ***me-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

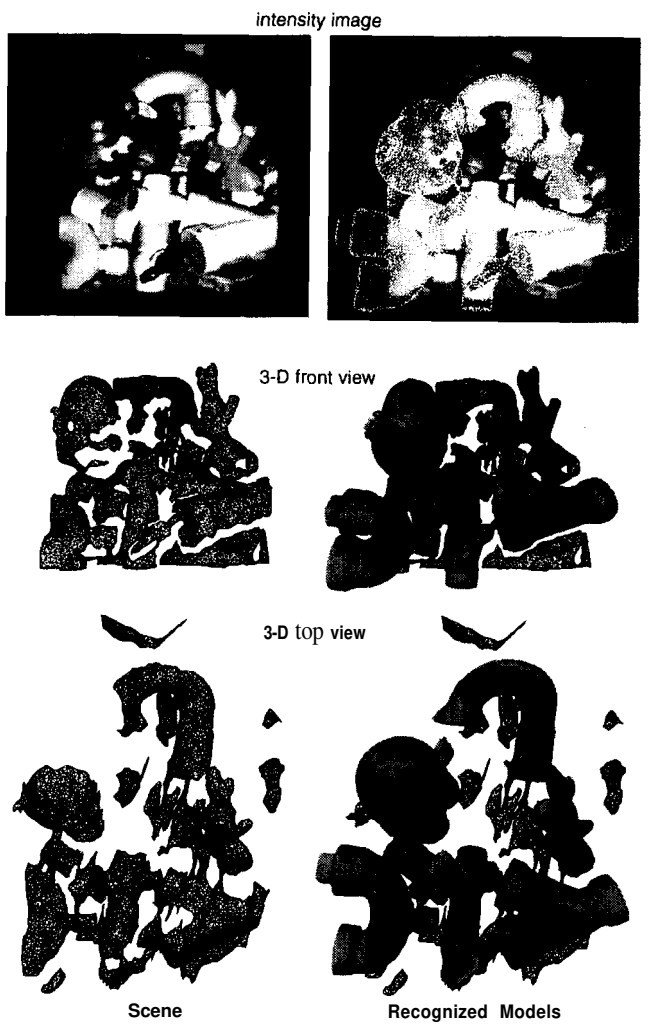

Figure 5. Simultaneous recognition of 7 models from a library of 20 models in a cluttered scene.

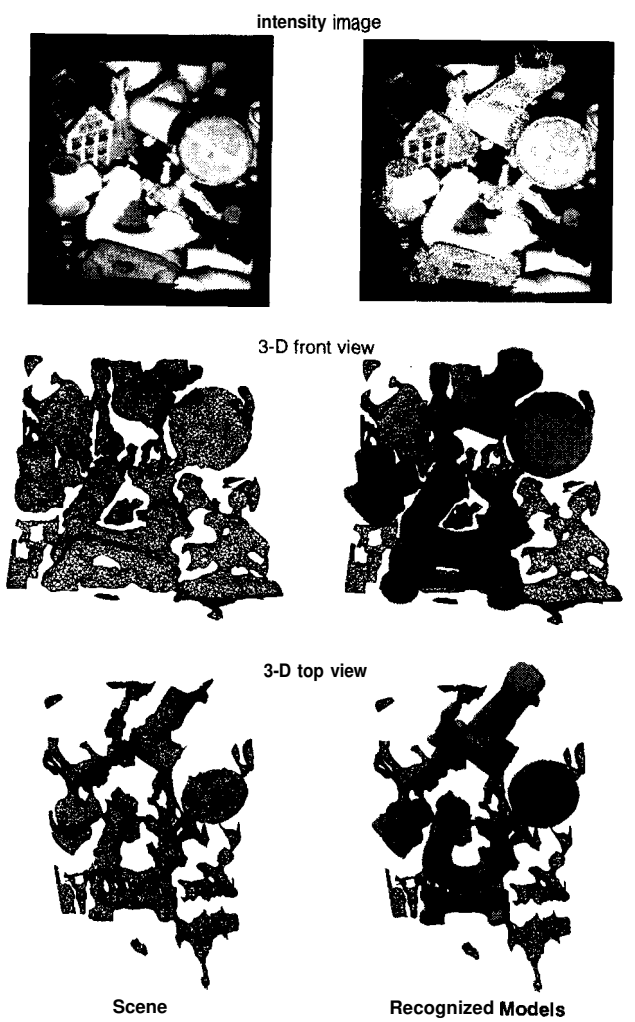

Figure 6. Simultaneous recognition of 6 models from a library of 20 models in a cluttered scene.

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

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

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

$$\text{clutter} = \frac{\text{clutter points in relevant volume}}{\text{total points in relevant volume}} \tag{7}$$

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 ail 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 ail 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 ail 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 triais.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. Briefly looking at both scatter plots shows

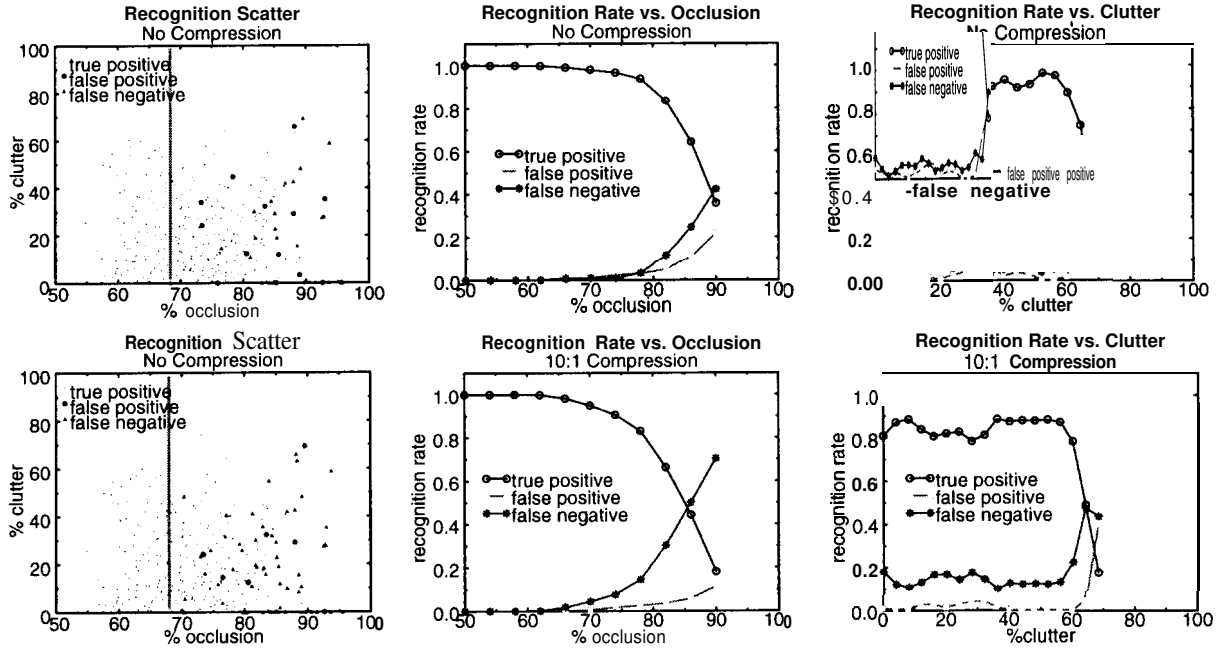

Figure 7. Recognition states vs. clutter and occlusion for compressed and uncompressed spin-images (left). Recognition state probability vs. occlusion for compressed and uncompressed spin-images (middle). Recognition state probability vs. clutter (right).

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 7, one notices that recognition rate is effected by occlusion. At low occlusion values, no recognition failures are reported, while at high occlusion values, recognition failures dominate. This indicates that recognition will almost always work if sufficient model surface area is visible. The decrease in recognition success after a fixed level of occlusion is reached (70%) indicates that spin-image matching does not work well when only a small portion of the model is visible. This is no surprise since spin-image descriptiveness comes from accumulation of surface area around a point. In 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 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. 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 [7].

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.

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] 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-l 12, 1993.

[3] 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-l 130, 1997.

[4] 0. 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.

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

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

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

[8] 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.

[9] 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.*

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

[11] 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.

[12] 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.

[13] 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.

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.