Large Data Sets and Confusing Scenes in 3-D surface Matching and Recognition

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

Abstract

In this paper, we report on recent extensions to a surface matching algorithm based on local 3-D signatures. This algorithm was previously shown to be effective in view registration of general surfaces and in object recognition from 3-D model data bases. We describe extensions to the basic matching algorithm which will enable it to address several challenging, and often overlooked, problems encountered with real data.

First, we describe extensions that allow us to deal with data sets with large variations in resolution and with large data sets for which computational efficiency is a major issue. The applicability of the enhanced matching algorithm is illustrated by an example application: the construction of large terrain maps and the construction of accurate 3-D models from unregistered views.

Second, we describe extensions that facilitate the use of 3-D object recognition in cases in which the scene contains a large amount of clutter (e.g., the object occupies 1% of the scene) and in which the scene presents a high degree of confusion (e.g., the model shape is close to other shapes in the scene.) Those last two extensions involve learning recognition strategies from the description of the model and from the performance of the recognition algorithm using Bayesian and memory-based learning techniques, respectively.

1. Introduction

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

Satisfying those requirements is notoriously difficult. Many approaches have been proposed in the past (e.g., [1][9][24][25], as a small sample); for each of them results are presented in controlled data, leaving many of the challenges faced in practical applications of those techniques unanswered. For example, many matching techniques cannot be scaled to very large data without combinatorial explosion; similarly, many matching techniques cannot handle scenes with large degree of clutter.

In this paper, we attempt to address some of the challenges met in practical applications through recent extensions and example applications of our approach to surface matching. Through those extensions and results, we will show that the matching technique can not only handle the standard controlled settings for recognition and view registration, but can also handle much more challenging cases such as data sets with large resolution variation, efficient processing of large data sets, clutter rejection for recognition in large scenes, and recognition in scenes with a high-level of confusion.

Let us first briefly recall our basic approach to surface matching. A. Johnson proposed in [13][14] an approach based on the computation of local surface signatures called spin-images. The signature of a given basis point $\boldsymbol{P}$[1] is computed by every projecting point $\boldsymbol{M}$ in a support region centered at $\boldsymbol{P}$ onto a two-dimensional coordinate system defined by the distance of $\boldsymbol{M}$ to the tangent plane, β, and its distance to the normal line, α. By histogramming the values of α and β computed in the support region, we obtain a two-dimensional image which is used as the signature at $\boldsymbol{P}$. Those signatures can be shown to be invariant to rigid transformations, resistant to occlusion and clutter, and easily computable from 3-D data sets with minimal requirements on the underlying shapes and the connectivity of the surfaces. In the rest of the paper, we will use the expressions "surface signatures" and "spin images" interchangeably.

Matching two surfaces (from two views, or from a model object and an observed surface) proceeds by first finding points on the two surface with similar signatures and by extracting a set of geometrically consistent matches from those initial matches. The similarity between signatures is computed from the correlation between the signatures. It can be shown that the signatures are discriminating enough to find a good set of initial matches, even with a high degree of clutter. Analysis and results for the basic signature-based matching algorithms for view registration and object recognition can be found in [18] and [17], respectively. Applications to industrial object recognition and to indoor mapping are reported in [16] and [4]. Those results show that the approach matching of general surfaces without segmentation or feature extraction, and without any prior knowledge of pose information.

Based on those initial results, we have explored ways to enhance the algorithm to make it more better suited to practical applications. First, because the signature computation works directly on point sets, it is potentially sensitive to variation in data resolution. For example, if the distribution of points varies differently on one sur-

1. An oriented point is specified by a 3-D location and an orientation and typically corresponds to a surface point and its normal.

face than on the other, the signatures may be substantially different. Our initial technique addressed this problem by resampling the data. We described a more principled and effective approach in Section 2.1.

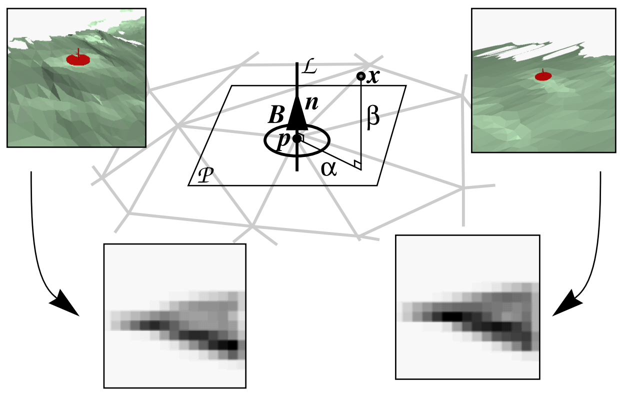

Figure 1. The signature at point p is computed by recording the distance of all nearby points x from the surface normal n (α) and the distance from x to p along n (β). Corresponding points from different views have similar signatures.

A second issue is the cost of computing the signatures. In particular, because the signature is computed in a support region around each basis point, it is critical to be able to visit the points in this support region without visiting the entire data set, a critical issue in the case of large data sets. We show in Section 2.2.how the signatures can be computed efficiently using standard geometric data structures.

We show an example of application of those surface matching techniques to the problem of *terrain map construction* (Section 2.3.) which involves the registration of large terrain maps from airborne or ground range sensors. Through this example, we show that the surface matching techniques can tolerate large variation of data resolution, that they can operate with the very large data sets that are typical of 100m+ terrain maps, and that they can deal with unstructured terrain. In this example, we address the worst case in which no prior knowledge on the relative poses between maps is known in advance.

The last set of extensions is in the context of *object recognition*. While the basic performance of object recognition from model databases is reported extensively in [17], we concentrate here on two cases in which object recognition becomes particularly challenging. First, we consider the cases in which the scene size is very large compared to the model size. This occurs, for example, if the task involves finding a single object in a large room environment. In that case, the problem becomes one of fast clutter rejection in order to avoid unacceptable computational burden. We describe in Section 3.1. an approach called *3-D shape cueing* for fast filtering of clutter points and show preliminary results[1].

The second situation to which we pay particular attention is the case in which the model shape is very similar to the shape of other objects in the scene. In that situation, the fundamental problem is to automatically modify the model, or the parameters of the recognition algorithm, such that the parts of the model that are most discriminating to the other objects are used in priority. We propose a *learning technique* for enhancing the performance of the object recognition algorithm and show preliminary results in Section 3.2. Interestingly, standard learning techniques can be used in this case because we use image-like signatures rather than structural descriptors. Such use of learning techniques in 3-D recognition is a new a promising direction of research.

2. Surface Matching with Large Data Sets

2.1. Variable Data Resolution

In the representation described thus far, the signature images are computed by histogramming the vertices of the model or scene meshes. As a result, the distribution of the vertices on the mesh directly affects the spin-images. In fact, two meshes with different vertex distributions may generate very different signatures at the same basis point. Therefore, in order for the surface matching algorithm to work properly, some constraint has to be enforced on the distribution of vertices on the meshes. Specifically, it can be shown that the spin-images remain stable as long as the vertices are uniformly distributed on the surface. In all the results presented thus far, a decimation algorithm was applied to all the meshes prior to matching in order to enforce this uniformity constraint. The decimation algorithm is described in detail in [15].

Although this approach works well in practice, it has several problems. First of all, there are cases in which the data simply cannot be made uniform without loosing a great deal information because the variation of resolution in the input sensor data is too large. A typical example is terrain data taken from a forward-looking sensor. The sensor data varies from high-resolution at close range to quadratically decreasing resolution as the range from the sensor increases. Variations in data point spacing of as much as 1:10 are routinely observed on terrain data. The second problem is that the uniform decimation requires on the same order of computation time as the matching itself, even though much faster decimation and filtering algorithms do exist [11]. Finally, as a guiding principle of this work, we attempt to make the matching algorithm as general as possible. In that respect, using arbitrary mesh distributions is critical.

The solution to those problems is to compute the signatures by integrating over the entire surface rather than by computing α and β values at the vertices only. Essentially, this requires interpolating the spin-image values "in between" the mesh vertices. The simplest way of achieving this is to raster scan each triangle of the mesh (Figure 2) and to compute the (α,β) coordinates of each point inside the face. The corresponding spin-image entry is incremented by a constant amount for each new

1. This topic is included for completeness and to serve has an introduction to the topic of learning for 3-D shape recognition. Therefore, we limit ourselves to a summary of the key results in that section. Additional results on cueing are reported in [5].

point. This algorithm can be made efficient by using a fast geometric test in order to determine whether a face is inside the region of influence of the basis point and is within the boundary of the spin-image space.

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

The second approach is exact in that it computes the spin-images by integration over the whole surface without additional sampling. In this approach, the boundary of each triangle is mapped into $\alpha\beta$-space, as shown in Figure 3 (a). Each edge of the face maps to a segment of the hyperbola. The hyperbolic segments computed in the projection are then used for determining which cells of the spin-image may contain some portion of the triangle. Figure 3 (b) shows the portion of the spin-image that contains a portion of the triangle based on the segments of Figure 3 (a). Finally, each cell in the spin-image is incremented by the area of the part of the triangular face that is mapped to that cell in $\alpha\beta$-space.

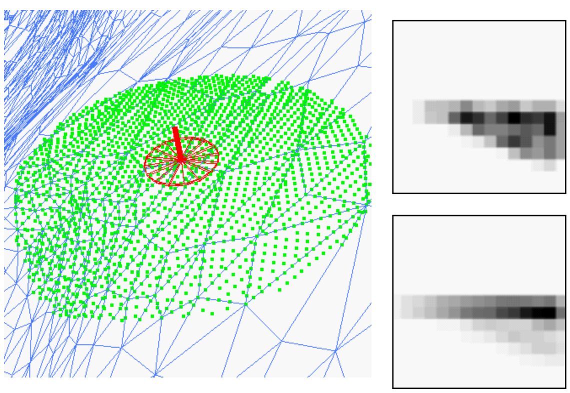

Figure 2. Discrete vertex interpolation: each face is raster-scanned before mapping to spin-image. The sample points used for the scanning are shown in green in the neighborhood of the basis point shown in red. The vertex-based signature for this point (top right) is degraded when compared with the corresponding face-based signature (bottom right).

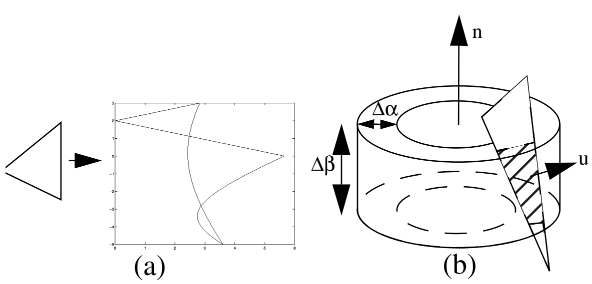

Figure 3. Continuous interpolation: each face is mapped to $\alpha\beta$ space by mapping its edges (a); cells inside the mapped region are incremented by the area of the intersection of the face and the volume in space corresponding to the cell (b).

This last step is illustrated geometrically in Figure 3 (b). The region of 3D corresponding to the cell (α,β) is an annulus of height $\Delta\beta$ and thickness $\Delta\alpha$. The cell is incremented by the surface area of the intersection between the triangle and this annulus.

Because it uses the actual surface area for incrementing the spin-image cells, this algorithm computes an "exact" mapping of the surface to the signatures, given a mesh discretization of the surface. A systematic comparison of the discrete and continuous approaches remains to be done, but the benefits of interpolation over vertex mapping have been demonstrated. Figure 4 shows two meshes of an object at two different resolutions and different point distributions. Given a basis point, the region of influence of which is shown on the mesh, the signatures computed from the only vertices are quite different as expected. However, the signatures computed using interpolation are similar.

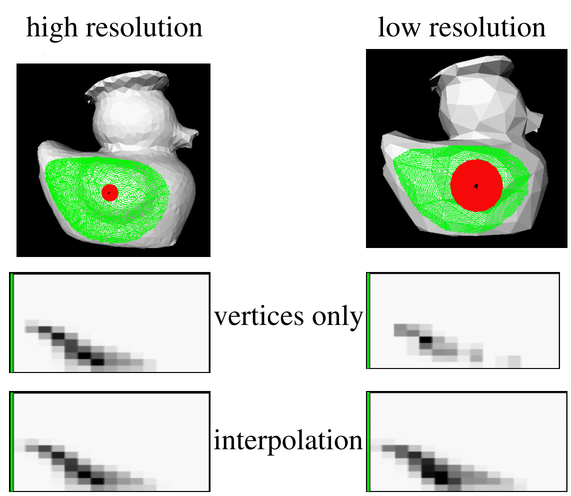

Figure 4. Comparison of the signatures at a basis point computed at different resolutions using the vertex and interpolation methods.

How much more similar are the signatures computed using interpolation? A partial answer to that question is shown in Figure 6. An object (a faucet) was discretized at three different resolutions, denoted by *r1*, *r2*, and *r3* using the algorithm of [11], i.e., no attempt was made to enforce uniformity of the point distribution. The number of vertices on the three versions of the object is: 1585, 57, and 120, respectively. A set of points $\{p_i\}$ was selected on the object independently of the discretization and, for each basis point, the similarity of the signatures $s(r, r', p_i)$ was computed using different pairs of resolutions r, $r' =$ *r1*, *r2*, and *r3*. The similarity is computed using the formula introduced in [14] and is close to zero for uncorrelated spin-images and has high values for similar images. Figure 6. shows the histograms of $s(r1, r2, p_i)$ and $s(r1, r3, p_i)$ using the vertex method and the interpolation method. The histograms show a substantial improvement in the similarity of signatures. In particular, most of the similarity values $s(r1, r3, p_i)$ are so low that such a difference in resolution makes matching impractical. Using the interpolation method, however, increases the similarity values to a level suitable for matching.

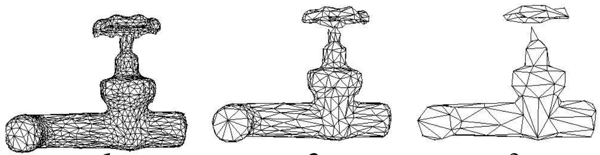

Figure 5. Test object for interpolation at three different resolutions.

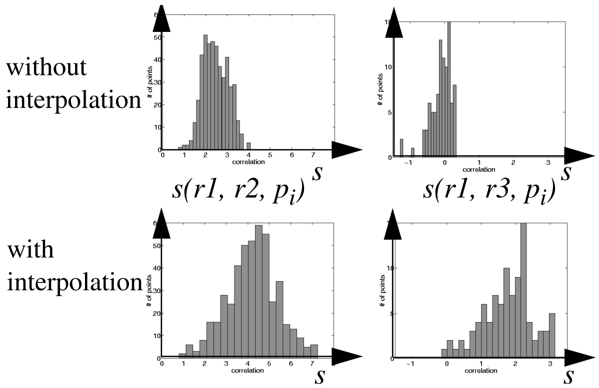

Figure 6. Histograms of similarity between meshes at different resolutions with and without interpolation.

This experiment shows that, by performing integration over the surface rather than resolution-dependent sampling, the interpolation method permits the comparison of surfaces at very different levels of resolution. The examples of Figure 4 and Figure 6 were obtained from controlled experiments; the question of the effectiveness of surface interpolation in real applications will be answered in the applications Section below.

2.2. Fast Access Data Structures

This part of the work addresses the practical use of the matching techniques, in particular using the more advanced surface integration, for the very large data sets that one expects to encounter in applications such as building terrain models, or virtual models of large interior environments.

The main potential obstacle to the practical use of signature techniques is that the computation of the signatures may become prohibitively expensive if the data set is very large. In particular, the amount of computation needed from each face in order to compute the signature at a given basis point is much more substantial than in the standard method. Therefore, it becomes particularly important that only the points that are inside the region of influence of a basis point be used for computing the corresponding signature. This amounts to designing a data structure which enables fast access to selected regions of the cloud of points representing the surface. The standard approach to this type of problem is the use of variants of the K-D tree structure designed for fast access in multidimensional spaces. After evaluation of several implementation of similar geometric data structures, the best design turned out to be a regular hierarchical data which is similar to octrees, except that, because we are working with 2-D manifolds, the tree is sparse and access can be efficiently implemented by a fast hashing method. The graphs below illustrate the improvement in signature computation speed obtained using this technique.

Because of the overhead involved in computing the data structure, and because of the overhead involved in computing the hashing function and retrieving points from the data structure, this technique is really beneficial only for large data sets. In fact, the computation is slower for data size of moderate size. The graphs show that the crossover point is at approximately 8000 points (Figure 7.) This technique should not be used for smaller data sets.

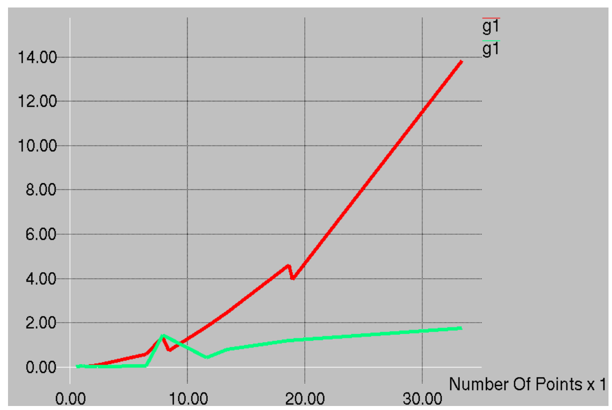

Figure 7. Computation time as a function of the number of points without indexing data structure (red), and with indexing data structure (green).

2.3. Example Applications

In the previous sections, we demonstrated the performance of the algorithms enhancements for variable resolution and large data sets on controlled test examples. The question of the effectiveness of the approach in practice still remains. An ideal for demonstrating the algorithm is the problem of building large, high-resolution three-dimensional (3-D) representations of unstructured terrain using terrestrial range sensors[1].

In this application, it is necessary to address the issues of widely varying resolution, absence of reliable features, and very large data sets. Furthermore, the data sets are typically very large, thus demonstrating the use of signature-based surface matching for large-scale problems.

---

1. Terrestrial sensors operate over a range of meters to hundreds of meters and are distinguished from object modeling sensors, which operate in a range of millimeters to meters, and remote sensors, which operate at kilometer range and higher. Terrestrial range sensors can be ground-based or airborne.

In this applications, the map registration component receives as input a pair of arbitrary polygonal surface meshes, possibly containing holes and disconnected patches, and outputs the 6-DOF rigid-body transform that best aligns the two meshes. In order to fully exercise the surface matching capabilities, we do not assume any prior knowledge of the transformation between individual maps. In a way, this is the testing the worst case for the matching algorithm. Techniques for taking advantage of prior knowledge of relative poses is described in [4].

In our experiments, we create meshes from real range data obtained from two sensors, one ground-based and one aerial. The Ben Franklin 2 (BF2), a scanning laser range finder mounted on Navlab 2 and Navlab 5, two of Carnegie Mellon University's (CMU's) autonomous ground vehicles, produces 360 degree by 30 degree range and reflectance images in a radius of 52 meters (Figure 8.)[12].

.

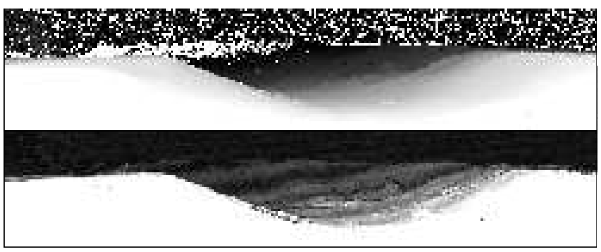

Figure 8. Example range (top) and reflectance (bottom) images acquired with the BF2 laser range finder.

The BF2 was set to generate range and reflectance images of size 6000 x 300, corresponding to angular resolutions of 0.06° and 0.1° in the horizontal and vertical directions respectively. The initial data is subsampled by a factor of 5 horizontally and 3 vertically and then converted to a mesh.

Once a mesh is formed, it is simplified to a predetermined number of faces using Garland's simplification algorithm [11]. A pair of simplified meshes is then presented to the registration engine. spin-images are generated for each point in one mesh and for a fixed percentage of points in the other. The signatures are computed using the discrete vertex interpolation technique described above in order to eliminate the effect of resolution variation. Candidate correspondences are found using the correlation-based similarity metric, and the best correspondences are grouped based on geometric consistency. We then determine the rigid-body transform that best aligns the correspondences from each group. The candidate transforms are then verified and ranked. We select the top match as our registration result. Finally, we use high-resolution versions of the surface meshes to refine our transform estimate using a modified version of the ICP algorithm. After all pairs of meshes in a sequence have been registered, they are integrated into a single global map using a standard voxel-based method.

Figure 10. shows an example of integrated map obtained using the first data from the BF2 scanner. The environment is a slag heap near Pittsburgh. The map was created by registering eleven individual maps obtained at 3 to 5 meters interval (the ground truth poses were not used in the registration.)

In this example, the support region for computing the spin image signature is 5mx5m. The initial data size is 1.8M points for the BF2 data, which is reduced to 15000 for matching. For that data, the variation of resolution from near to far range is greater than 1:10 due to the shallow incidence angle of the laser at long range.

In addition to ground-based data, the technique has been applied to data from the CMU autonomous helicopter [21], collected both from local sites and from the Haughton crater in the Canadian Artic, with similar performance. For reasons of space, the resulting images cannot be included here, but sample results are available at [26]. Those results show that the matching technique performs well with large data sets and that it can tolerate large variations in resolution. Two issues that are critical to the practical application of surface matching.

3. Object Recognition in Large and Confusing Scenes

3.1. Shape Cueing

A major obstacle to the use of 3-D recognition technique in realistic settings is that the object to be recognized occupies typically a fairly small portion of the scene data. For example, if one were to try to recognize an object in a scan of an entire room, the size of the data set actually measured on the object would only be a small fraction of the total data size. For any recognition algorithm, this situation translates into spending most of the time examining clutter points. Many algorithms may not even be able to work at all because of combinatorial explosion. It is therefore critical that efficient ways of reducing the clutter be designed.

In our case, points are randomly selected in the scene as basis points for comparison with the basis points of the model. An obvious enhancement to the matching algorithm would be a mechanism by which a large percentage of the points in the scene could be filtered out. We call this mechanism "cueing," in that it suggests possible areas of the scene where the object might be located. The matching engine can confirm or reject points in the suggested areas. We briefly describe below the approach chosen to cueing.

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

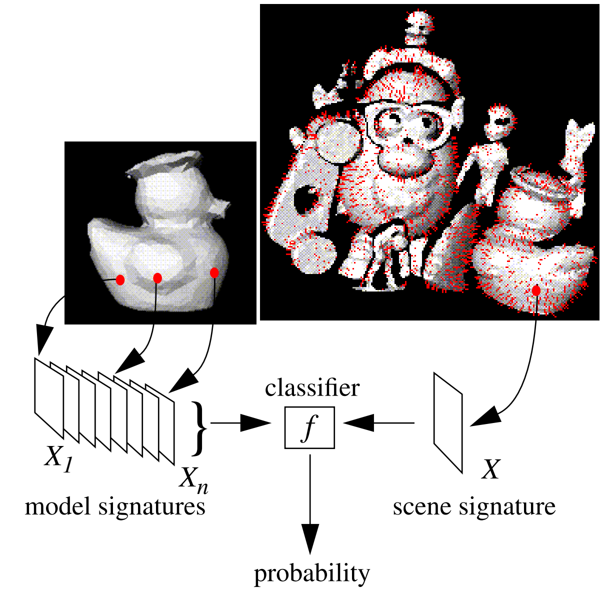

Figure 9. General approach to cueing: the complete set of signatures on the model (shown on the left) is summarized into a single classifier which is applied to the signatures computed at the scene points.

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

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

To find scene points which have the highest probability of belonging to a model, we need to find the MAP hypotheses $i \rightarrow A$; that is, the point i which maximize $P\langle i \rightarrow A | X_i \rangle$. No data is known about the prior probability that any particular point i may be found on a model, and no data is known a priori about the probability of a particular spin image X_i occurring, so uniform distributions $P\langle i \rightarrow A \rangle$ and $P\langle X_i \rangle$ are assumed. This reduces the problem to finding $max_i \; P(X_i | i \rightarrow A)$ or the maximum-likelihood hypothesis of $i \rightarrow A$ given X_i over all i's. Because of the computational cost of evaluating the actual joint probability distribution between the spin image bins, we assume conditional independence between the pixels in X_i, using the standard "naive

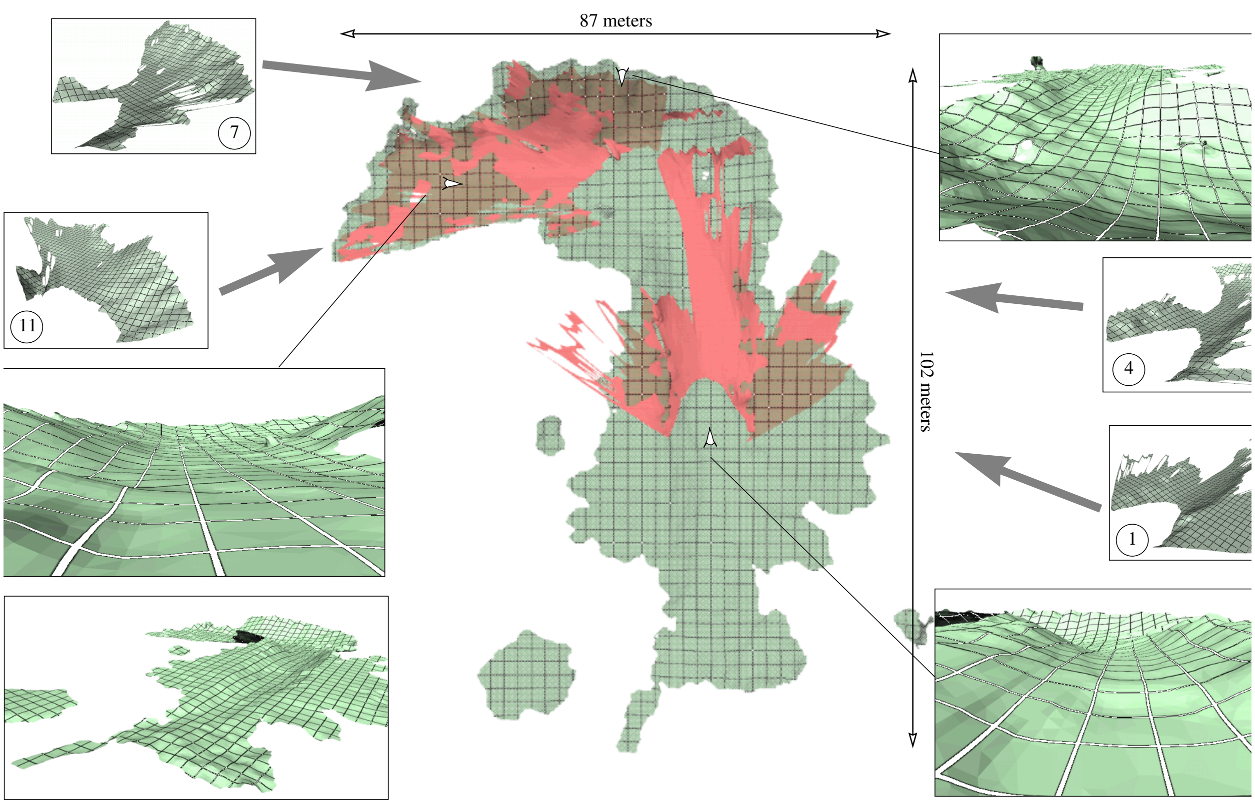

Figure 10. Top view of the integrated terrain map for the eleven data sets in the mesa sequence (center). The numbered insets illustrate individual data sets, and two sets (1 and 11) are overlaid on the top-view. The larger insets show various perspective views of the map, and the white arrows indicate the location and direction of the viewpoint. Grid lines are two-meters apart on the combined map and one meter apart on the individual data sets.

learning" approximation [8]; that is, if $X_i(\alpha,\beta)$ denotes the value of bin (α,β) in X_i, then we assume

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

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

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

This method of point classification was first tested on simple controlled scenes in which the model object is combined with other objects in scenes of increasing complexity. Figure 11. shows such an example using an elbow joint as the model. The points selected by the classifier are shown as dots overlaid on the data. Most of the points are correctly selected on the object in this example, even though the scene contains several other objects. The performance of the classifier is evaluated by taking the ratio of the percentage of points correctly classified by the classifier to the percentage of the scene that really consisted of model points. Formally, if N_m is the number of model points in the scene according to ground truth, N_c is the number of clutter points in the scene, N'_m is the number of scene points that were classified as model points and actually were points on the model, and N'_c is the number of scene points classified as model points, but were really points lying somewhere on the clutter, then we define the performance of the classifier by the ratio $r = \rho'/\rho$, where $\rho' = N'_m/(N'_c + N'_m)$ and $\rho = N_m/(N_c + N_m)$. A large value for r indicates a reduction in the number of points that are considered for matching. If the filtering were perfect, all the points would lie on the model, therefore, $r = 1/\rho$ is the maximum value of r.

This approach is particularly attractive for scenes in which the object model occupies a small fraction of the scene. We have tested the algorithms on scenes with up to 99% clutter. The results are summarized in Figure 12. in which the ratio of selected points r is plotted against the ratio of model size vs. scene size, ρ. The results show that the classifier performed better than random selection ($\rho = 1$) in all cases, and often close to the optimal ($r = 1/\rho$.) Obviously, the performance of the classifier degrades in scenes containing multiple instances of very similar surfaces. However, it never makes the initial random selection worse, i.e., ρ' is always greater than ρ.

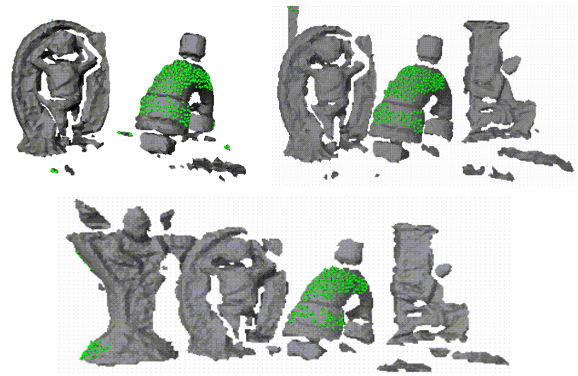

Figure 11. 3-D cueing examples: the points retained as part of the U-joint with high probability are shown as dots overlaid on the surfaces.

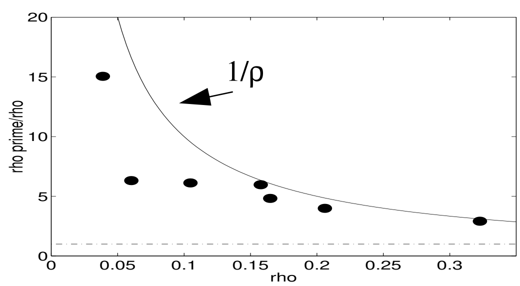

Figure 12. Cueing performance.

3.2. Learning for Clutter Rejection and Model Discrimination

Another obstacle to good performance of object recognition in practice is the fact that many surfaces in the scene may be similar to large portion of the model, thus leading to high degree of confusion in the scene. Unless extremely careful parameter adjustments are applied, a recognition program would typically be confused because it has no way a-priori to know that some part of the model can be easily confused with parts of the scene. This issue is often overlooked in two ways. First, example scenes are often selected so that the model is not very similar to objects in the background. Second, the recognition program may be manually tuned so that the model can still be discriminated reliably.

The fundamental issue is that the parts of the data that can lead to confusion must be given a lower weight in the matching. The problem is that this extremely difficult to do manually, and it is next to impossible to do automatically in the absence of any example of "confusing" scenes.

The alternative is to use a set of typical confusing scenes to automatically train the recognition algorithm so that, once it is trained, it is able to discard from future scenes the potentially confusing parts. More precisely, given a scene $S = \{p_i\}$ and a model M, we want to assign to each point pi a weight w_i which represent the degree to which this point may be useful for matching. The policy by which the w_i's are assigned is learned from the performance of the recognition algorithms on prior example scenes S_j.

This approach is actually a generalization of the approach taken for the 3-D cueing. In cueing, we have "learned" from the distribution of signatures on the model a policy by which we assign a weight to points in the scene reflecting their likelihood of belonging to the model. In the present case, we assign weights based on past performance of the recognition algorithm.

Because it is difficult to derive a probability distributions in signature space in general, we use a memory-based learning approach to the training phase. Let p_i^j be point i of scene j of the training set and s_i^j be its signature. We divide the set of points in the training scenes into those that are correctly matched to points on the models, the set of which is denoted by **+** for "positive examples", and those that are incorrectly matched, the set of which is denoted by **-** for "negative examples". The **-** set is essentially the set of scene signatures that can be easily confused with signatures on the model.

After collecting the **+** and **-** sets, w_i is computed for a new point p_i, from a new test scene, with signature si by retrieving the signatures from **+** and **-** that are closest to s_i, the sets of which are denoted by $V^+(s_i)$ and $V^-(s_i)$, respectively. The weight is then computed by:

$$w_i = \sum_{s \text{ in } V^+(s_i)} 1/d(s,s_i) \; - \sum_{s \text{ in } V^-(s_i)} 1/d(s,s_i)$$

In this formula, d is the distance in signature space. The weights are normalized to compensate for the variability in the density of examples in the signature space. Intuitively, this formula says that signatures that have been found to be confusing in the past are given a low weight. Point with low weight are then discarded when comparing the signatures for recognition. If one were to try to do manual adjustment of the parameters of the matching algorithm, one would follow essentially the same steps: identify those pieces of surfaces that are confusing and set the appropriate threshold to discard them. The learning algorithm does this automatically.

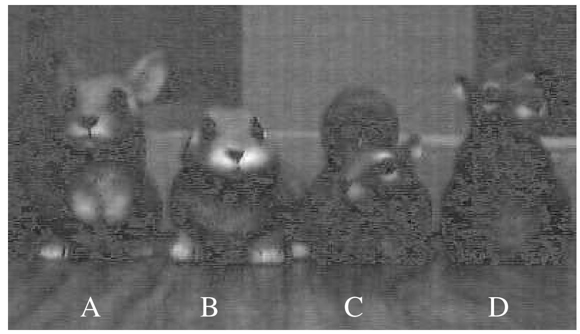

Figure 13. Example objects for learning from recognition examples.

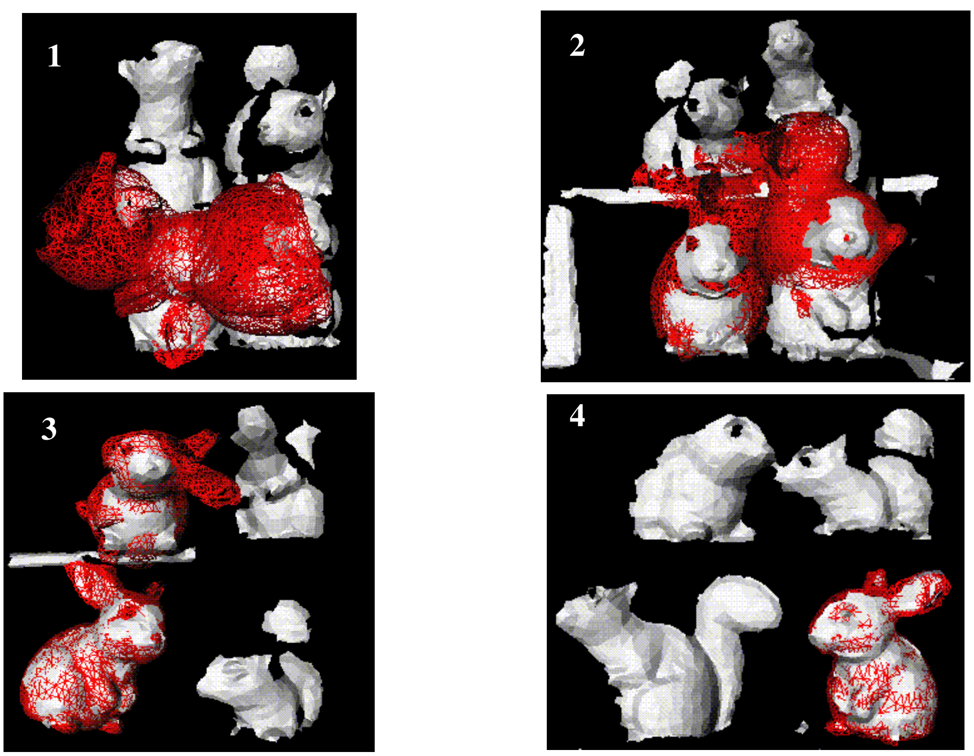

Figure 14. Enhancement of recognition performance in confusing scenes through learning (see text.)

An example illustrating the learning concept is shown in Figure 13. and Figure 14. Figure 13. shows the pictures of several objects labeled A to D. Object A is used as the reference model against which the scene points are matched, the other objects are placed in the scene at random positions. The objects B to D are purposefully chosen to be similar so as to create a large degree of confusion in the scene. In fact, A and B have mostly the same shape (two bunnies!) except for a few areas (the ears and the front paws area.) Many of the surface on C and D are also similar to large portion of A. In such an extreme case, it would be near impossible to tweak a recognition algorithm in order to reliably discriminate between the two objects.

Figure 14. shows a sequence of scenes, each of which contains one instance of the model and copies of the other objects in random orientation. The scenes are labelled 1 to 4 in order in which they are fed to the algorithm. Starting with no information, i.e., using the default matching parameters, the recognition gets confused on scene 1 and returns wrong matches. The model is displayed in wireframe in the poses found by the recognition program and overlaid on the scene data displayed using shaded surfaces.

Note that, in order to have a fair comparison, a fixed threshold on recognition is used so that multiple recognized instances may be reported. Furthermore, no further verification is applied to the reported matches. This correctly simulates the situation in which the recognition program starts in a default configuration which is updated after training.

After placing the data from scene 1 in the training set, the recognition algorithm is exercised on scene 2, a new scene. The algorithm still performs poorly because the training data has not reduced the confusion level sufficiently. The performance improves when two images are used for training as shown on the result of image 3. In particular, bunny A is still confused with B. After three scenes are included in the training set, the algorithm converges to a good policy for selecting the weights and the recognition is perfect as shown by the result on a new test scene, number 4.

There are several technical difficulties in implementing this approach. Most importantly, we are working in signature space which may be of prohibitively large dimensionality. For example, the typical signatures used in this paper are 30x30 which would force us to index in a 900-dimensional space! This problem is addressed by first projecting the signatures onto a lower-dimensional space spanned by the principal components of the signatures. This technique, described in [17], was originally designed for recognition of large library of objects and was shown to be effective at compressing the signature by taking advantage of the redundant information in the signatures of a given object. The results shown in Figure 14. were obtained by projecting on the seven most significant signature components. For those models, the seven directions provides model reconstruction with 80% accuracy. A reconstruction of 99% can be achieved with 27 components. The performance of the learning algorithm are similar in that case.

The second difficulty is to retrieve V^+ and V^- in an efficient manner. We typically limit the size of the *V*s to the k nearest neighbors (k = 5 in the example shown here), which reduces the problem to finding the k-nearest neighbor problem. This problem has been extensively studied in the field of memory-based learning and we are using the standard data structures for efficient data access in memory-based learning. A related problem, also typical of memory-based learning, is the issue of data accumulation. Unlike other learning schemes, memory-based learning keeps accumulating the training data, which may lead to unacceptable performance when the training set becomes very large. The solution is to collapse similar observations, i.e. similar signatures s_i^j from the training set into single observations.

4. Conclusion

We have presented enhancements to a generic 3-D surface matching algorithms which directly address the issues of large data sets and confusing scenes. Those issues which are not normally addressed are critical applying 3-D recognition techniques in practice. In the area of large data sets, we need to identify the limitations of the algorithms. The example on terrain maps shows that it is possible to perform matching of large data sets. The limit beyond which matching is no longer practical needs to be determined. In the area of "confusing scenes", the use of learning techniques for dynamic improvement of recognition performance is an exciting area of investigation. In particular, it may be practical to use those techniques for adaptation of a recognition algorithm to different applications, i.e., to different classes of scenes.

Acknowledgements

This research was supported in part by NSF Grant IRI-9711853 and by ONR Grant N00014-95-1-0591.

References

[1] A.P. Ashbrook, R.B. Fisher, C. Robertson, and N. Werghi. Finding Surface Correspondences for Object Recognition Using Pairwise Geometric Histograms. *Proc. European Conference on Computer Vision*. 1998.

[2] R. Bergevin, D. Laurendeau, and D. Poussart. *Registering Range Views of Multipart Objects*. Computer Vision and Image Understanding. Vol. 61, No. 1, pp. 1-16. 1995/

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

[5] O. Carmichael and M. Hebert. 3-D Cueing: A Data Filter For Object Recognition. Proc. Intl. Conference on Robotics and Automation. 1999.

[6] C.S. Cheng, Y.P. Hung, and J.B. Chung. A Fast Automatic Method for Registration of Partially

Overlapping Range Images. *Proc. International Conference on Computer Vision*. 1998.

[7] 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.

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

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

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

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

[12] J. Hancock, D. Langer, M. Hebert, R. Sullivan, D. Ingimarson, E. Hoffman, M. Mettenleitner, C. Froehlich. Active Laser Radar for High Performance Measurements, in *Proc. IEEE International Conference on Robotics and Automation*, pp. 1465-70, 1998.

[13] A. Johnson and M. Hebert. Object Recognition by Matching Oriented Points. Proceedings IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, June 1997.

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

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

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

[17] A. Johnson and M. Hebert. Efficient Multiple Model Recogition in Cluttered 3-D Scenes. Proceedings IEEE Conference on Computer Vision and Pattern Recognition, Santa Barbara, June 1998. To appear in IEEE Trans. PAMI, 1999.

[18] A. Johnson and M. Hebert. View Registration by Matching Oriented Points. Proceedings International Conference on Recent Advances in 3-D Digital Imaging and Modeling, Ottawa, May 1997.

[19] 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.

[20] D. Laurendeau, G. Roth, and L. Borgeat. Optimisation Algorithms for Range Image Registration. *Proc. Vision Interface*. 1996

[21] R. Miller and O. Amidi. 3-D site mapping with the CMU autonomous helicopter, in *Proc. Intl. Conf. on Intelligent Autonomous Systems* (IAS-5), June, 1998.

[22] C. Olson and L. Matthies. Maximum likelihood rover localization by matching range maps. In *Proc. IEEE International Conference on Robotics and Automation*, pp. 272-277, 1998.

[23] 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.

[24] 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.

[25] 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.

[26] http://www.cs.cmu.edu/~dhuber/mapping/building_maps.html