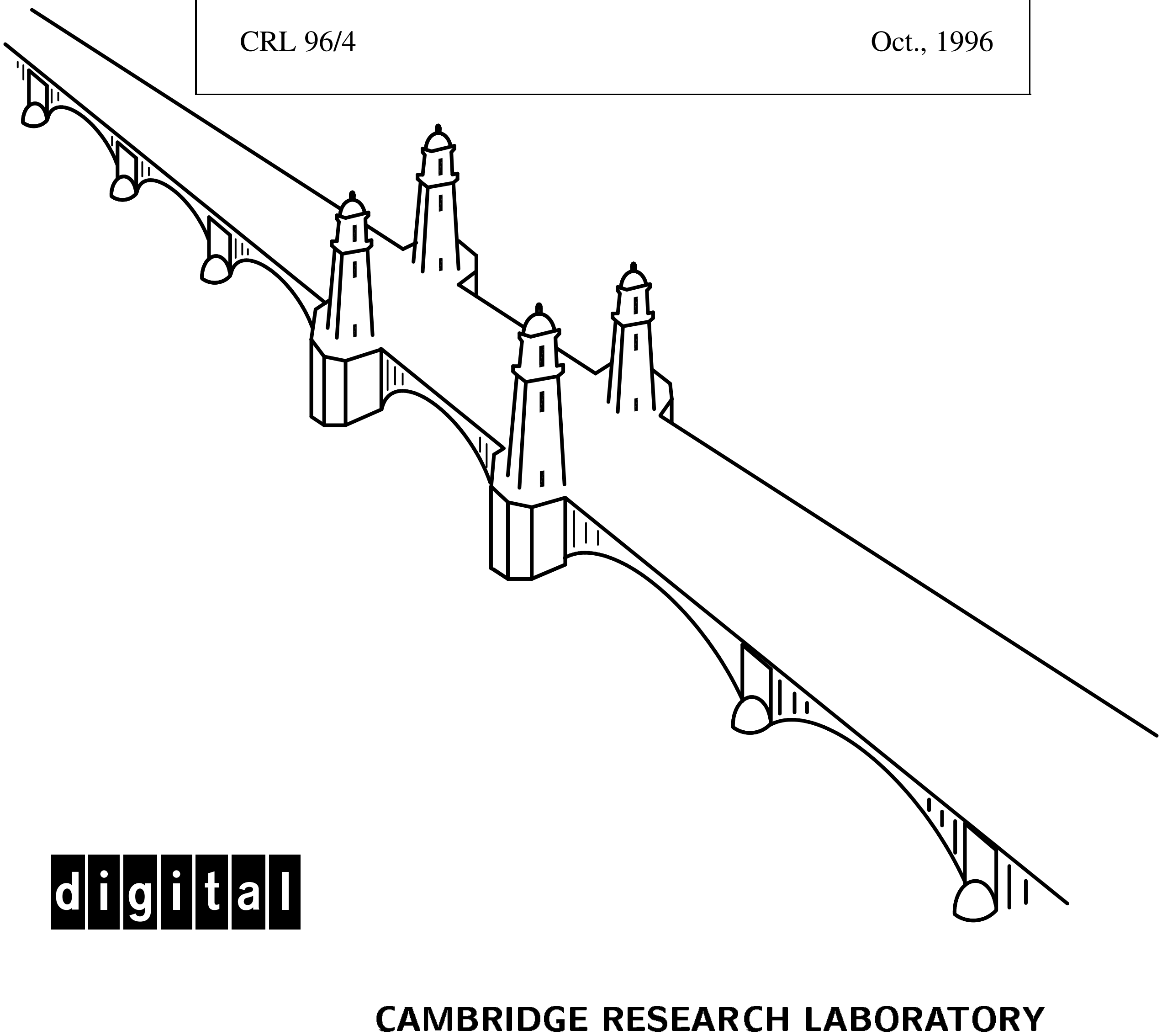
Registration and Integration of Textured 3-D Data

Andrew Johnson and Sing Bing Kang

Digital Equipment Corporation

Cambridge Research Lab

CRL 96/4 Oct., 1996



CAMBRIDGE RESEARCH LABORATORY

Technical Report Series

Cambridge Research Laboratory
One Kendall Square
Cambridge, Massachusetts 02139

Registration and Integration of Textured 3-D Data

Andrew Johnson[1] and Sing Bing Kang

Digital Equipment Corporation

Cambridge Research Lab

CRL 96/4 Oct., 1996

Abstract

In general, multiple views are required to create a complete 3-D model of an object or a multi-roomed indoor scene. In this work, we address the problem of merging multiple *textured 3-D data sets*, each of which corresponding to a different view of a scene or object. There are two steps to the merging process: registration and integration.

Registration is the process by which data sets are brought into alignment. To this end, we use a modified version of the Iterative Closest Point algorithm (ICP); our version, which we call *color ICP*, considers not only 3-D information, but color as well. This has shown to have resulted in improved performance.

Once the 3-D data sets have been registered, we then integrate them to produce a seamless, composite 3-D textured model. Our approach to integration uses a 3-D occupancy grid to represent likelihood of spatial occupancy through voting. The occupancy grid representation allows the incorporation of sensor modeling. The surface of the merged model is recovered by detecting ridges in the occupancy grid, and subsequently polygonized using the standard Marching Cubes algorithm. Another important component of the integration step is the texture merging; this is accomplished by trilinear interpolation of overlapping textures corresponding to the original contributing data sets. We present results of experiments involving synthetic and real scenes.

[1]The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213

Keywords: 3-D scene modeling, 3-D registration, 3-D data merging, color iterative closest point (ICP), 3-D occupancy grid technique.

Contents

List of Figures

List of Tables

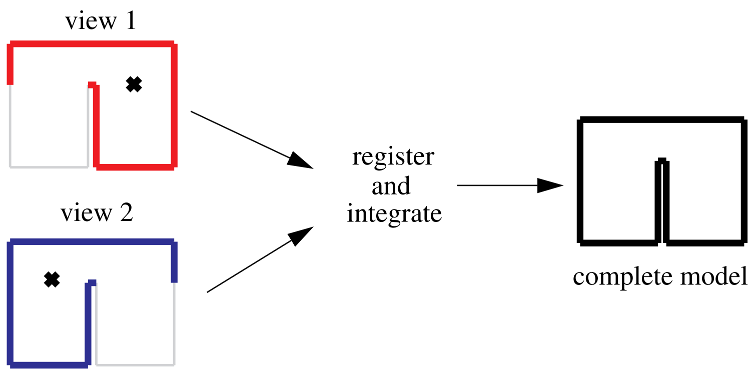


Figure 1: Illustration of the merging problem. The thick $\times$'s mark the locations of the camera, with each corresponding to a different range data set.

1 Introduction

There is an increasing interest in modeling scenes for virtual reality applications, either in the areas of business (real estate, architecture, information-dispensing kiosk), education (electronic museums and multimedia books), or entertainment (interactive 3-D games, movies). The option of creating virtual environments by capturing real scenes through video cameras is getting particular attention, given the labor-intensive and thus expensive nature of creating models by hand using a 3-D geometric modeler. The problem of creating models of a large scene or an entire object is that any given view of the camera or a depth imaging such as a light-stripe rangefinder is insufficient—thus merging of multiple views taken at different locations is usually necessary. This is then followed by integrating the different views to result in a seamless 3-D textured model. The problem is illustrated in Figure 1.

A lot of research work has been done in the area of model creation through multiple view merging. Shum *et al.* [Shum *et al.*, 1994], for example, recover the merged model through simultaneous determination of planar surface parameter location and pose of constituent range data sets. They assume, however, that the surfaces of objects can be represented using planar patches. There is work that involves modeling of free-form (i.e., smooth-varying) objects as well [Higuchi *et al.*, 1993; Shum *et al.*, 1995], but they require exhaustive search in pose space to determine relative camera location.

The act of reconstructing the model of a real scene from a camera images (as opposed to direct

3-D data recovery from rangefinders) has been termed *VideoCopying*[1]. (This term is an allusion to "photocopying" in 2-D.) The idea is to be able to reconstruct and model the 3-D world by merely using a commercially available video camcorder and continuously videotaping the scene while moving the camcorder. The structure of the scene can be recovered from multiple images using structure from motion algorithms such as [Azarbayejani and Pentland, 1995; Szeliski and Kang, 1994; Taylor and Kriegman, 1995] if the camera motion is not known.

1.1 Outline

This document is organized as follows: The rest of this section briefly describes how 3-D data that are used in this work is recovered, and defines the notion of *textured 3-D data*. The steps involved in our proposed 3-D data merging work are depicted in Figure 2. The first step in the merging process is data set registration; this step is described in Section 2. Here we introduce the idea of using color in addition to 3-D location in the registration step.

Subsequent to registration is the integration step to produce a seamless 3-D textured model. Section 3 delineates the integration step, which involves the use of occupancy grids based on sensor modeling and ridge detection to recover composite 3-D surfaces. The technique to blend textures from different data sets is also explained in this section.

Section 4 provides some implementational details of the technique of merging multiple textured 3-D data sets. Discussion of data merging issues and future work is given in Section 5 before we summarize our work in Section 6.

1.2 Recovery of 3-D scene data

In our work, we use 3-D data recovered from omnidirectional multibaseline stereo, i.e., using multiple panoramic images [Kang and Szeliski, 1996]. Each panoramic image spans a 360° horizontal field of view. The primary advantage of this method is that at any given camera center location, the scene can be recovered at a very wide horizontal field of view. This is done without resorting to any intermediate 3-D merging.

The omnidirectional multibaseline stereo approach to recover 3-D data and subsequently the scene model is summarized in Figure 3. We provide only a brief outline of the approach here.

[1]The term "VideoCopying" was coined by S.B. Kang in 1995, and was originally referred to in the web site http://www.research.digital.com/CRL/personal/sbk/research/scene-sensing.html.

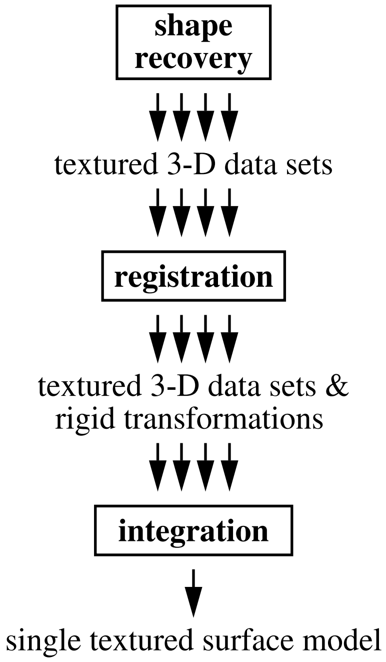


Figure 2: Processes involved in our approach for merging multiple 3-D data sets.

Figure 3: Generating scene model from multiple 360° panoramic images.

Full details can be found in [Kang and Szeliski, 1996]. The approach is straightforward: at each camera location in the scene, sequences of images are captured while rotating the camera about the vertical axis passing through the camera optical center. Each set of images are then composited to produce panoramas at each camera location. The stereo algorithm is then used to extract 3-D data of the scene. Finally, the scene is modeled using the 3-D data input; the model is rendered with the texture provided by the 2-D image input.

1.3 Textured 3-D data

As described earlier, omnidirectional multi-baseline stereo produces a set of 3-D points with associated 2-D image coordinates in a panoramic image. By connecting the 3-D points based on the connectivity given by the Delaunay triangulation of the 2-D image coordinates, a 3-D triangular surface mesh is generated [Kang *et al.*, 1995]. The surface normal at each vertex in the mesh can be determined by fitting a plane to the vertex and all of the vertices adjacent to it in the mesh. The surface normal is then set to the normal of the best fit plane that points toward the sensor origin.

Every point on the surface mesh (including any point on the faces of the mesh) can be projected into the panoramic image to obtain the color for that point. In this manner, the texture from the panoramic image can be mapped onto the surface mesh. A surface mesh and associated image that texture maps it is called a textured 3-D data set. The color for a point on the surface mesh of a textured 3-D data set is determined by projecting the 3-D position of the point (x, y, z) into the panoramic image (of height H_I and width W_I) to determine its 2-D image coordinates (u, v); the set of equations governing this projection is given by (2).

$$
\begin{aligned}
\theta &= \tan^{-1}\left(\frac{y}{x}\right) \\
\phi &= \sin^{-1}\frac{z}{\sqrt{x^2+y^2+z^2}} \\
u &= (2\pi-\theta)\frac{W_I}{2\pi} \\
v &= \frac{H_I}{2} - \frac{W_I}{2\pi}\tan\phi
\end{aligned}
\tag{1}
$$

Since the projected point will not necessarily project to an exact integer pixel location, the color of the point is determined from bilinear interpolation of the colors from the four pixels surrounding the 2-D image coordinates. The purpose of bilinear interpolation is to smoothly resample the image when the point does not project to an exact integer pixel location. We call a 3-D point with its associated color a *color-point*.

2 Registration

Registration is the process by which two data sets are brought into alignment. In the case of 3-D modeling from images, we are interested in determining the rigid transformation that aligns two textured 3-D data sets, so that they can be placed in a common world coordinate system. Since no assumptions can be made about the shape of the data, the registration algorithm used must be able to handle free-form surfaces. The Iterative Closest Point algorithm (ICP) [Besl and McKay, 1992; Zhang, 1994] is an established algorithm for registration of free-form surfaces that is simple to implement and easy to modify to meet specific needs. A requirement of the Iterative Closest Point algorithm is that the two data sets to be registered are coarsely aligned. Since we have an initial guess for the transformation that aligns two data sets (based on the coarse measurements of relative camera placements), we can use an ICP algorithm to register textured 3-D data sets.

2.1 Iterative closest point algorithm

Registration of free-form surface is a hard problem because it is difficult to establish correspondences between data sets. To solve this problem Besl and McKay [Besl and McKay, 1992] proposed the Iterative Closest Point algorithm which establishes correspondences between data sets by matching points in one data set to the closest points in the other data set. Traditional ICP works as follows. Given a point set M and a surface S: For each m_i in M, find s_i, the closest point on the surface S. Next the rigid transformation T that minimizes the distance between the (m_i, s_i) pairs in a least squares sense is calculated. All of the points in M are transformed by T, and the process is repeated until the distance between closest points falls below a threshold d_{max}. A pseudo-code description of the algorithm is given in Figure 4. ICP is an elegant way to register free-form surfaces because it is intuitive and simple. Besl and McKay's algorithm requires an initial transformation that places the two data sets in approximate registration and operates under the condition that one data set be a proper subset of the other. Since their algorithm looks for a corresponding scene point for every model point, incorrect registration can occur when a model point does not have a

```
ICP(point_set M, surface S) {
   while (d(T)>d_max) {
      for each m_i in M {
         s_i = ClosestPoint(m_i,S)
      }
      transformation T = min_T d(T) = min_T Σ_i ||s_i − T(m_i)||^2
      M = TransformPointSet(M,T)
   }
}
```

$$T = \min_T d(T) = \min_T \sum_i \left\| s_i - T(m_i) \right\|^2$$

Figure 4: Pseudo-code of traditional ICP algorithm.

corresponding scene point due to occlusion in the scene.

Zhang [Zhang, 1994] also proposed an iterative closest point algorithm that has two improvements over the algorithm of Besl and McKay. The first improvement used k-dimensional trees [Friedman *et al.*, 1977; Sproull, 1991] to speed up the closest point computation. The second improvement uses robust statistics to generate a dynamic distance threshold on the distance allowed between closest points. This dynamic distance threshold is used to relax the requirement that one data set be a proper subset of the other, so that partially overlapping data sets can be registered. He showed good results with stereo data, which motivated our use of the ICP algorithm for registration.

Simon *et al.* [Simon *et al.*, 1994] created a real time 3-D tracking system that built on Besl and McKay's ICP algorithm. They added many improvements to the algorithm to increase the speed of registration including k-d trees for closest point computation, closest point caching, storage of 2-D face representations and decoupled acceleration of the ICP algorithm. They found that the greatest speed improvements were due to the use of k-d trees and the ICP acceleration.

We have developed an ICP algorithm that builds on the algorithm presented by Zhang [Zhang, 1994]. In addition to using k-d trees for closest point computations and a dynamic distance threshold, our algorithm uses shape and color information to improve the registration beyond that obtained with an ICP algorithm that uses just shape information.

2.2 Color ICP

During integration of textured 3-D data, shape as well as texture are integrated to form the final consensus surface model. Our approach to texture integration is to project the texture from all of the registered data sets onto the final consensus surface where the overlapping textures are blended. For texture to be blended correctly, the texture projected from all of the data sets must be accurately aligned on the final consensus surface. In other words, for correct alignment of texture, registration on the order of a few image pixels projected into the scene is required. For example, a 2000 pixel wide panorama becomes misregistered by one pixel if the estimated rotation is incorrect by 0.18 degrees. Inaccuracies in scene shape introduced by the shape recovery algorithm (omnidirectional stereo) are too large to obtain the accuracy in registration needed to blend texture using a traditional ICP algorithm. However, by including color in the closest point computation of the ICP algorithm, the necessary registration accuracy can be obtained.

In traditional ICP, closest points are searched for in 3-D Euclidean space, so two data sets are registered based on similarity in shape. However, for registration of textured 3-D data, accurate alignment of shape and texture is required. This can be accomplished by modifying the distance metric used to compute closest points to include a measure of texture similarity. This idea is shown in Figure 5. Since texture is conveyed by the color projected onto the points in the surface mesh, adding a measure of color difference to the Euclidean distance metric will be sufficient. Consider two points $\mathbf{p}_1$ and $\mathbf{p}_2$ with positions $\mathbf{x}_1 = (x_{11}, x_{12}, x_{13})$ and $\mathbf{x}_2 = (x_{21}, x_{22}, x_{23})$ and colors $\mathbf{c}_1 = (c_{11}, c_{12}, c_{13})$ and $\mathbf{c}_2 = (c_{21}, c_{22}, c_{23})$ then the 6-D L_2 color/shape distance between the points is

$$\begin{aligned} d_6(\mathbf{p}_1, \mathbf{p}_2) \quad = \quad & \Big[(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + (x_{13} - x_{23})^2 + \\ & \alpha_1(c_{11} - c_{21})^2 + \alpha_2(c_{12} - c_{22})^2 + \alpha_3(c_{13} - c_{23})^2\Big]^{\frac{1}{2}} \end{aligned} \tag{2}$$

where $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ are scale factors that weigh the importance of color against the importance of shape. These scale factors and the color model used will be discussed later in this section. Adding color to the distance metric used to compute closest points will ensure that points that are close to each other and of similar color are aligned. Then end result will be better registration than can be obtained with shape alone.

In general, with stereo data, the number of recovered 3-D points is much smaller than the

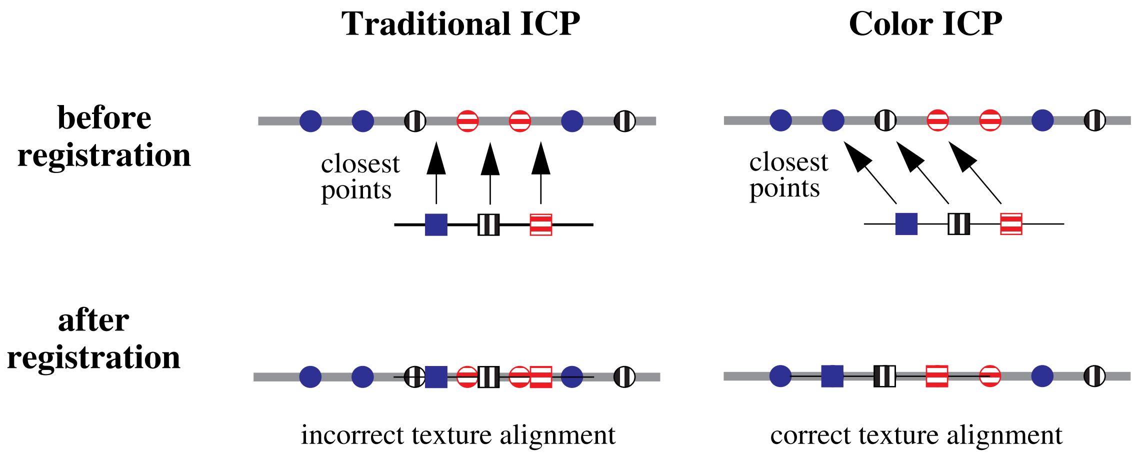


Figure 5: Demonstration of the use of color in registration. In traditional ICP closest points depend only on shape, so it can produce incorrect texture registration. Since closest points depend on color and shape in Color ICP, it will aligns texture correctly.

number of pixels in the image. This can be caused by many things including lack of texture variation in the image, occlusions and subsampling to reduce the amount of processing. Therefore, the color variation in the image will not be completely captured by the color projected onto the vertices of the surface mesh. To adequately capture the color variation in the image, we super-sample the surface mesh by creating extra 3-D points on the faces of the surface mesh which can be projected into the image to obtain their color. The extra points for each face are created on a regular grid of fixed size attached to each face. We set the size of the grid so that the number of 3-D points is between one fifth and one tenth the number of pixels in the image. For registration it is only necessary to super-sample one of the data sets because the super sampled set will contain all of the color points in the other data set.

The flow of the *Color ICP* algorithm (whose pseudo-code is shown in Figure 6) is similar to the ICP algorithm developed by Zhang. Suppose that two textured 3-D data sets M and S are to be registered. First, super sample S, as detailed above, to create a dense set of color-points representing the shape and texture of S. Next, create a set of color-points from the vertices of the surface mesh of M and transform the points of M by the initial guess of the registration transformation. Then using the distance metric from (2), create a 6-D k-D tree for efficient determination of the closest color-point in S. Once the data structures are initialized, the iterations of the ICP algorithm

```
ColorICP(int Dmax, transformation IT, textured_3D_data_set M,
      textured_3D_data_set S) {
   color_point_set CS = CreateSuperSampledColorPoints(S)
   color_point set CM = CreateColorPointsFromVertices(M)
   tree TS = CreateClosestPointTree(CS)
   CM = TransformPointSet(CM,IT)
   while ( !StoppingCriteriaMet(T) ) {              // Equation 4
      color_point_set US = ClearPointSet()
      color_point_set UM = ClearPointSet()
      for each m_i in CM {
         s_i = ClosestColorPoint(m_i,TS,CS)
      }
      if (ColorDistance(m_i,s_i) < Dmax ){          // Equation 2
         AddColorPoint(US,s_i)
         AddColorPoint(UM,m_i)
      }
      transformation T := BestTransformation(US,UM)// Equation 3
      CM = TransformPointSet(CM,T)
      Dmax = UpdateDistanceThreshold(US,UM,T)      // Zhang's method
   }
}
```

Figure 6: Pseudo-code of color ICP algorithm.

commence. For each color-point $\mathbf{m}_i$ in M find the closest color-point $\mathbf{s}_i$ in S using the 6-D tree for S. Given the $(\mathbf{m}_i, \mathbf{s}_i)$ pairs, compute the rigid transformation T that will minimize the 3-D Euclidean distance between them using the quaternion method of Faugeras and Hebert [Faugeras and Hebert, 1986]:

$$T(t_x, t_y, t_z, r_x, r_y, r_z) = \min_T \sum_i ||\mathbf{s}_i - T(\mathbf{m}_i)||^2 \tag{3}$$

Finally, transform the points in M by T repeat the iterations until the convergence criterion is met. A pseudo-code description of the Color ICP algorithm is given in Figure 4.

To make our ICP algorithm robust to registration when the model is not a proper subset of the scene, we have incorporated the dynamic maximum distance threshold employed by Zhang [Zhang, 1994]. This threshold limits the maximum distance between closest points (6-D); if two points are farther apart than this threshold, they are not used to compute the rigid transformation. Zhang applies rules based on the statistics of the histogram of distances between closest points in order to set this threshold automatically. We use all of the same rules for setting the distance threshold, except we do not use the rule that sets the threshold when the registration is very bad. Instead of finding the first minimum after the main peak in the histogram, and setting the threshold to this if the registration is very bad, we apply a simpler rule that sets the distance threshold to its starting value.

Our stopping criterion is met when the magnitude of the translation and the magnitude of a vector made from the rotation angles is fall below separate thresholds.

$$\begin{aligned} d_t &= \sqrt{t_x^2 + t_y^2 + t_z^2} < H_t \\ d_r &= \sqrt{r_x^2 + r_y^2 + r_z^2} < H_r \end{aligned} \tag{4}$$

By separating the stopping criterion into translational and rotational components, we have more control over the convergence of the registration. In particular, the Color ICP algorithm will continue to iterate if either the translational or rotational components of the computed transformation are significant.

If the 3-D position and color of points are to be compared then the scale that relates them must be determined. Before this scale can be determined, an appropriate color model that determines the color coordinates of a point must be chosen. Color is being used to register two textured 3-D data sets that may be created under different lighting conditions or with different sensors. Under normal lighting (i.e., white light) most variations in color of an object taken from different viewpoints will come from variations in shading. Shading generally affects the intensity of light coming from an object, but not its intrinsic color. Therefore, we would like the color model to separate intensity from intrinsic color, so that the role of intensity in the matching of color can be reduced. The color model that we chose that meets this criterion is the YIQ color model [Foley *et al.*, 1990]. In the YIQ model the intensity of light is conveyed by the Y channel and the intrinsic color (hue, saturation) is conveyed by the I and Q channels. The HSB color space also separates out the intensity of color, but its polar nature creates singularities which make the calculation of distance more complicated. The mapping from YIQ to the color coordinates of (2) is $(y, i, q) = (c_1, c_2, c_3)$.

The scale of color with respect to 3-D position is determined by the a vector in 8. To reduce

registration algorithm	transformation parameters						errors	
	t_x	t_y	t_z	r_x	r_y	r_z	E_t	E_r
ICP	1.976	0.806	-0.043	-0.380	0.196	1.112	0.050	1.191
Color ICP	1.993	0.793	-0.005	-0.049	-0.035	-0.018	0.011	0.041
Correct	2.000	0.800	0.000	0.000	0.000	0.000	0.000	0.000

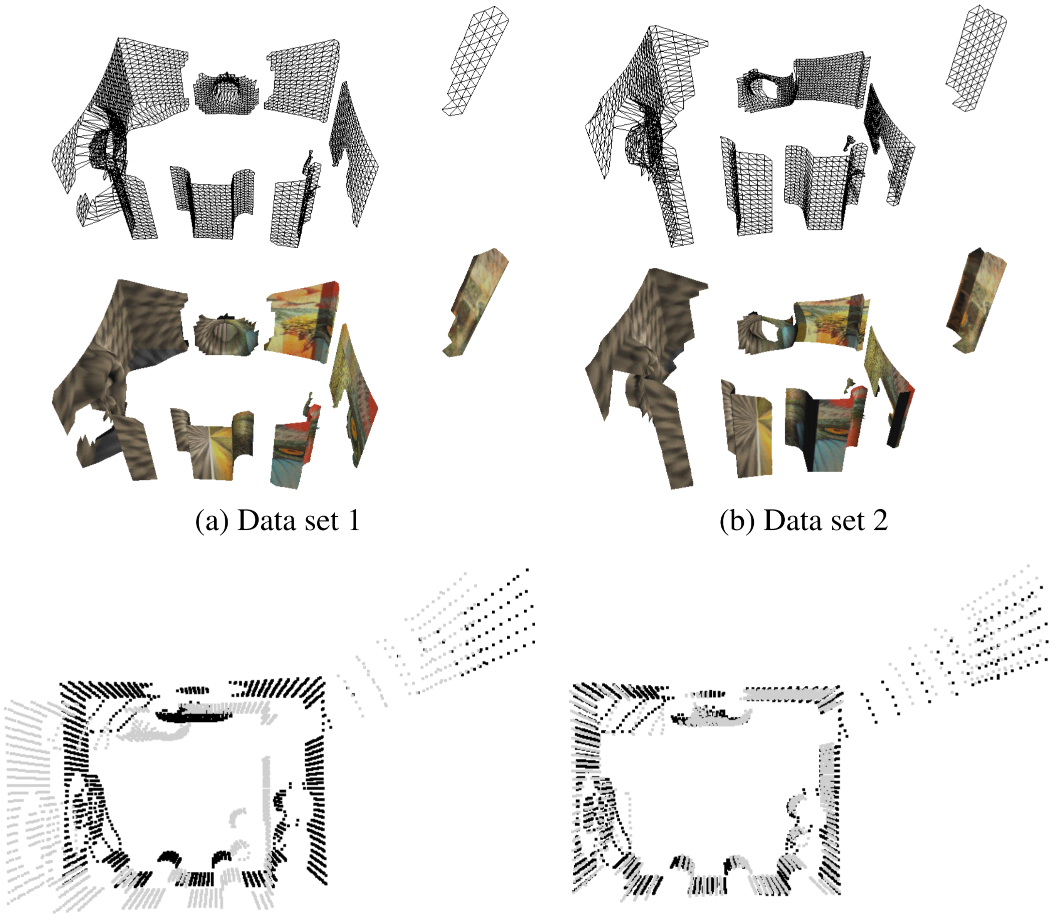
Table 1: Comparison of registration results to ground truth for a typical trial for the synthetic room. The angles are in degrees.

the effect of intensity on the matching of points, we make the scale of the Y channel one tenth the scale of the I and Q channels. We have produced excellent registration results when $\alpha = (1, 10, 10)$. Since the spatial variation of the 3-D textured data sets that we are merging is on order of 10 units, this scale factor makes the intrinsic color of points have an effect that is on order of the effect of the spatial coordinates.

2.3 Results

An example registration result is shown in Figure 7. At the top of the figure are shown two textured 3-D data sets (in wireframe and texture mapped) generated from a synthetic room model generated using the Rayshade modeling package [Kolb, 1994]. The room has 4 walls, a doorway into another room and various objects along the wall (tori, vases, columns). The walls of the room are also texture mapped with common vision images (such as the mandrill face image) to add extra texture to the scene. The room model is sufficiently complicated to test the Color ICP algorithm, while also allowing a comparison to ground truth since the exact transformation between the two data sets is known. At the bottom of Figure 7 are shown the points in the data sets before and after registration by the Color ICP algorithm. No misregistration is apparent.

Since we know the transformation between the data sets a comparison of the results of the algorithm to ground truth and traditional ICP can be made. Table 1 shows the transformation parameters $(t_x, t_y, t_z, r_x, r_y, r_z)$ calculated for traditional ICP and Color ICP in comparison with ground truth. Also shown are the translational errors (E_t) and rotational errors (E_r) between the computed registrations and ground truth. The Color ICP algorithm performs much better than the traditional ICP algorithm as can be seen from a comparison of their errors. As mentioned



(a) Data set 1 (b) Data set 2

(c) Before registration (d) After registration

Figure 7: Two textured 3-D data sets of a synthetically generated room shown in wireframe and texture mapped (top). A top view of the points in the sets before and after registration by the Color ICP algorithm (bottom).

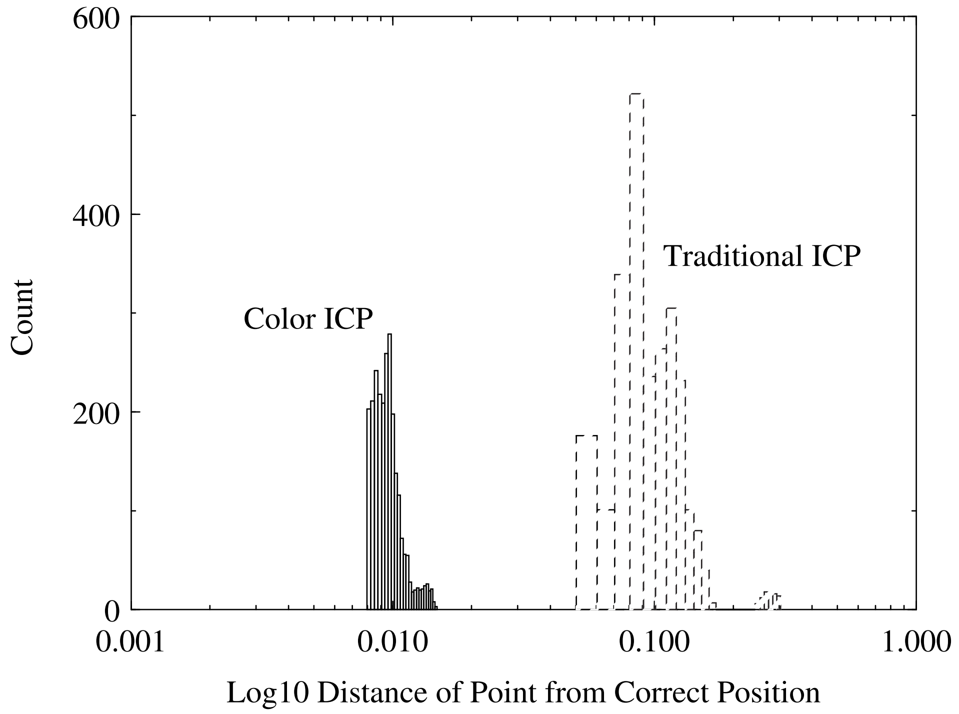


Figure 8: Histogram of registration errors for the traditional ICP and Color ICP algorithms from a typical trial (synthetic room case). The histogram clearly shows that the Color ICP algorithm is an order of magnitude improvement over the traditional ICP algorithm.

previously, to integrate texture, the rotational registration error should be on order of tenths of a degree. The registration produced by the Color ICP algorithm is within the error bound, so it can be used to register textured 3-D data sets for integration.

Another measure of registration error is the distance between a point after registration and the true location of the point. A histogram of this distance, for all of the points in the second textured 3-D data set, is shown for the traditional ICP and Color ICP algorithm in Figure 8. The median of the errors for the traditional ICP algorithm is around 0.10 and the median of the errors for the Color ICP algorithm is around 0.01, so the Color ICP algorithm is an order of magnitude better than the traditional ICP algorithm at registering textured 3-D data sets. Other measurements of registration error and an extensive discussion of the ICP algorithm can be found in [Simon, 1996].

3 Integration

The purpose of registering the individual textured 3-D data sets is to place them in a common world coordinate system. Once in the same coordinate system, the data sets can be combined into a single surface model that contains shape as well as appearance information. The process which combines textured 3-D data sets is called integration. Stated succinctly, the purpose of integration is to combine multiple co-registered 3-D data sets into a single textured surface model.

Our approach to integration is related to work in 3-D occupancy grids and volumetric approaches to integration.

3.1 Related work

The integration problem is an active area of research where the common approaches are divided into two groups based on the type of data input into the algorithm. The first group integrates unstructured point sets. The second group of algorithms are supplied structured data which provides some knowledge about the underlying surface shape usually in the form of a surface mesh. The structured data approaches can be broken down further into surface based and volumetric approaches.

Integration algorithms that can be applied to unstructured point sets are useful when no underlying surface information is available. The surface is constructed using proximity information in 3-D space. Boissonnat [Boissonnat, 1984] developed an algorithm for efficient computation of the Delaunay tetrahedronization of space. Veltkamp [Veltkamp, 1991] creates surfaces from unorganized points by generalizing the concept of closest point using a γ-neighborhood graph, when constructing a 3-D tetrahedronization of space. Hoppe *et al.* [Hoppe *et al.*, 1992] use an augmented Euclidean Minimal Spanning Tree to create a signed distance function from a set of unorganized points. They then polygonize the signed distance function using the Marching Cubes surface polygonizer. Bajaj *et al.* [Bajaj *et al.*, 1995] use alpha-shapes and Bernstein-Bezier forms to construct smooth surfaces from a set of unorganized points. Because unstructured point algorithms have no surface information to begin with, they produce smooth surfaces which can give unreliable surface estimates near discontinuities in the scene. Furthermore, these algorithms assume that the surface from a single object is to be recovered, making them less useful for integrating views of complex scenes.

The next group of algorithms assumes that some information describing the shape of the surface to be reconstructed is available. Usually this information is conveyed by connectivity information obtained through the data acquisition process (e.g., scanning). With connectivity, the surface normal at each point can be calculated, giving a richer description of the shape of the object than 3-D points without surface normals.

Surface based algorithms for integration of structured points usually operate on polygonal meshes. Soucy and Laurendeau [Soucy and Laurendeau, 1992] partition the points into disjoint sets based on a Venn diagram of views. Within each disjoint set they create a rectangular grid of surface points which are integrated along boundaries in the Venn diagram. Turk and Levoy [Turk and Levoy, 1994] developed a method which zips together overlapping surface meshes followed by adjustment of mesh vertex positions based on all the overlapping data. The algorithm of Rutishauser *et al.* [Rutishauser *et al.*, 1994] use a sensor error model to combine redundant points followed by a retriangulation step. By using the surface information these algorithms will produce better results than those produced by the unorganized point algorithms. However, dependence on a view-based retriangulation step will result in poor results near complex regions of high curvature. Chen and Medioni [Chen and Medioni, 1994] avoid the view dependent retriangulation step by growing a deformable surface to the surface data. However, their approach assumes the object being modeled is genus zero which is not true when modeling complex scenes.

The final group of integration algorithms constructs a continuous 3-D implicit function describing the surface using a volumetric data structure to discretely sample the function. Once the implicit surface is constructed, it is polygonized using the Marching Cubes algorithm to create the surface from the volumetric data. The methods vary in how the implicit surface is constructed and the volumetric data is organized. Hilton *et al.* [Hilton *et al.*, 1996] and Curless and Levoy [Curless and Levoy, 1996] have developed volumetric integration algorithms that construct a weighted signed distance function to the surface from structured point data. Hilton *et al.* use surface normal and distance to compute the signed distance function. Curless and Levoy augment their algorithm with a space carving step to clean up the meshes produced by polygonization. However, both of these methods are designed for modeling single objects without texture; in addition, prior accurate alignment of data sets is assumed.

Our algorithm is most similar to the volumetric approaches that construct a 3-D implicit surface function. We construct an implicit function describing the surface using a volumetric data structure. However, we approach the problem from the direction of probabilistic occupancy grids developed by Elfes [Elfes, 1987]. Occupancy grids describe the probability of surface based on the proximity of points and a sensor error model. The occupancy grid paradigm is intuitive and is easily changed to accommodate different sensors. Unfortunately, occupancy grids do not address the problem of surface extraction which is generally a difficult and error prone operation. In the next sections, we show how we can build a volumetric surface probability field and robustly extract a single consensus surface from it.

3.2 Occupancy grids

The occupancy grid paradigm was first developed to create detailed spatial maps from wide-angle sonar measurements [Eberly *et al.*, 1994]. Later it was determined that occupancy grids were a useful method for accumulating the probability of surface from multiple (possibly of different type) noisy sensors [Martin and Moravec, 1996]. The fundamental procedure for creating an occupancy grid is simple. First the world is partitioned into a fixed grid; in our case, the grid is a 3-D array of voxels. Stored in each voxel is the probability of surface existing in that voxel. When a surface point measurement is taken, the probability of surface is increased around that point according to a sensor model. After all of the reading have been taken the most likely surface will correspond to the ridge of probability in the 3-D array of voxels. The occupancy grid paradigm is very attractive for integration of 3-D textured data sets because it is

- *incremental*: Occupancy grids can be built gradually as sensor data is taken, so at any moment in the data collection process, the surface can be extracted. When additional data is collected, it can be inserted into the occupancy grid and a new surface generated. Incremental algorithms are important when a scene is gradually being explored.

- *simple*: The concept of accumulating surface evidence is intuitive and easy to implement.

- *free-form*: Occupancy grids make no assumption about the shape of the scene that is being measured. Since no model is assumed, any complex scene (up to voxel resolution) can be described.

- *flexible*: The occupancy grid paradigm can be modified easily to incorporate data from different sensors and sensing algorithms with different sensor error models. For example, occupancy grids can combine stereo point sets with spline-based structure-from-motion depth maps to create a single consensus surface. Occupancy grids can also be used to combine different sensing modalities. In our implementation, we use an occupancy grid to integrate surface shape as well as texture.

3.3 Sensor model

Before accumulating surface evidence in an occupancy grid, a sensor model must be determined. Our sensor model combines a model describing the sensor error distribution and a model that spreads the contribution of the point along the surface that is being imaged. Matthies and Shafer [Matthies and Shafer, 1987] showed that a good approximation of the error model for stereo is an ellipsoidal gaussian distribution centered at the measured 3-D point and oriented along the line of sight. Analytically, the sensor error model has the form of a cylindrically symmetric gaussian with its axis aligned with the local viewing direction

$$
\begin{aligned}
G_E(\alpha,\beta,\sigma_\alpha,\sigma_\beta) &= \frac{1}{\sqrt{2\pi}}\sqrt{\frac{1}{\sigma_\alpha^2}+\frac{1}{\sigma_\beta^2}}\exp\left(-\frac{1}{2}\left(\frac{\alpha^2}{\sigma_\alpha}+\frac{\beta^2}{\sigma_\beta}\right)\right) \\
\alpha &= \sqrt{\mathbf{x}\cdot\mathbf{x}-\mathbf{x}\cdot\hat{\mathbf{v}}} \\
\beta &= \mathbf{x}\cdot\hat{\mathbf{v}} \\
\mathbf{x} &= \mathbf{Q}-\mathbf{P} \\
\hat{\mathbf{v}} &= \frac{\mathbf{S}-\mathbf{P}}{||\mathbf{S}-\mathbf{P}||}
\end{aligned}
\tag{5}
$$

where α is the distance of the query point $\mathbf{x}$ from the unit viewing vector $\hat{\mathbf{v}}$ and β is the distance of the query point $\mathbf{x}$ along the unit viewing vector. The spread of the gaussian can be characterized by two parameters, σ_α^2 the variance perpendicular to the viewing direction and σ_β^2 the variance along the viewing direction. A 2-D slice of the sensor error geometry is given in Figure 9.

Matthies and Shafer show that the variances of the sensor error model should vary depending on the position of the sensed point. To reduce the amount of calculation per point, we have assumed that the variances of the sensor error model are fixed for all points. However, the variances of the model are set automatically by analyzing local changes in distance from the sensor. Consider a point $\mathbf{P}$ from surface mesh M that has N_M points and sensor origin $\mathbf{S}$. Call the local surface mesh neighborhood of $\mathbf{P}$ (points connected to $\mathbf{P}$ by the mesh), L_P with N_P points. The RMS spread in distance d_{rms} is calculated as follows:

$$\begin{aligned}
\bar{d} &= \frac{1}{N} \sum_{\mathbf{P} \in L_P} ||\mathbf{P} - \mathbf{S}|| \\
d_{\mathbf{P}} &= \frac{1}{N_P} \sum_{\mathbf{P} \in L_P} \bar{d} - ||\mathbf{P} - \mathbf{S}|| \\
d_{rms} &= \frac{1}{N_M} \sqrt{\sum_{\mathbf{P} \in M} d_{\mathbf{P}}^2}
\end{aligned} \tag{6}$$

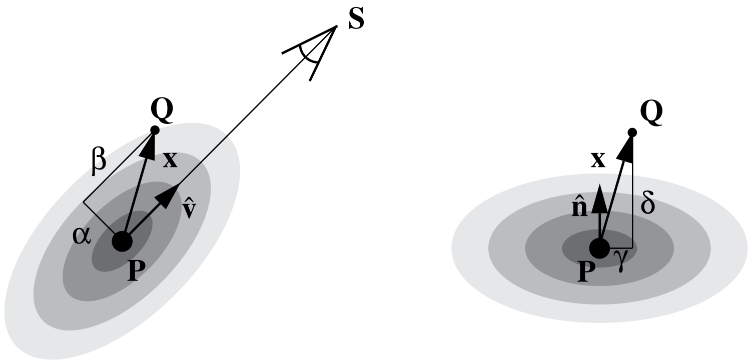
d_{rms} measures the average local change in distance which is a good measure of sensor error assuming that neighborhoods are locally planar, with normals roughly oriented along the viewing direction. The variances in the sensor error model are set automatically based estimated error as $\sigma_\alpha = d_{rms}$ and $\sigma_\beta = 2d_{rms}$.

Stereo returns discrete point measurements on the surface of objects. By spreading the contribution of a point along the tangent plane of the point, a continuous surface can be generated. To meet this end, a point spread function is added to the sensor model. The point spread function has the form of a cylindrically symmetric gaussian with its axis aligned with the local surface normal

$$\begin{aligned}
G_{\mathbf{S}}(\gamma, \delta, \sigma_\gamma, \sigma_\delta) &= \frac{1}{\sqrt{2\pi}} \sqrt{\frac{1}{\sigma_\gamma^2} + \frac{1}{\sigma_\delta^2}} \exp\left(-\frac{1}{2}\left(\frac{\gamma^2}{\sigma_\gamma} + \frac{\delta^2}{\sigma_\delta}\right)\right) \\
\gamma &= \sqrt{\mathbf{x} \cdot \mathbf{x} - \mathbf{x} \cdot \hat{\mathbf{n}}} \\
\delta &= \mathbf{x} \cdot \hat{\mathbf{n}} \\
\mathbf{x} &= \mathbf{Q} - \mathbf{P}
\end{aligned} \tag{7}$$

where γ is the distance of the query point x from the unit surface normal n̂ and δ is the distance of the query point x along the unit surface normal. The spread of the gaussian can be characterized by two parameters, σ_γ^2 the variance along the tangent plane and σ_δ^2 the variance along the surface normal. A 2-D slice of the surface spreading geometry is given in Figure 9.

The variances of the point spread function can be calculated automatically for each surface mesh by estimating the local resolution at each point. Ideally the variances of the spread function would be different for each point in the surface mesh, since the local resolution changes for each point. However, to reduce the computation for each point, the variances are fixed for each surface mesh and are based on the average mesh resolution for all of the points in the mesh. The average



Sensor Error Model G_E Point Spread Model G_S

Figure 9: Geometry for the two components of the sensor model: the Sensor Error Model, a cylindrical gaussian oriented along the sensor viewing direction and the Point Spread Model, a cylindrical gaussian oriented along the surface normal.

mesh resolution r_{av} for a surface mesh M with N_M points is

$$r_{av} = \frac{1}{N_M}\left(\sum_{\mathbf{p}_i,\mathbf{p}_j \in M} ||\mathbf{p}_i - \mathbf{p}_j||\right) \tag{8}$$

Based on the average mesh resolution, the variances of the point spread function can be set as $\sigma_\gamma = 2r_{av}$ and $\sigma_\delta = r_{av}$.

A linear combination is used to combine the sensor error and surface spread models into one sensor model M.

$$M(\lambda, \sigma_\alpha, \sigma_\beta, \sigma_\gamma, \sigma_\delta) = \lambda G_E(\sigma_\alpha, \sigma_\beta) + (1-\lambda) G_S(\sigma_\gamma, \sigma_\delta) \tag{9}$$

By adjusting the parameter λ on the interval [0,1], the relative importance of the sensor error and point spread models can be set. Convolution of the point spread model with the sensor error model is a more rigorous way of combining the models, but computationally we found it infeasible because both models change dynamically with the point being processed.

3.4 Surface probability field

The voxels of a traditional occupancy grid store a single scalar value, the probability of surface. Given structured data a more sophisticated occupancy grid representation can be used that encodes

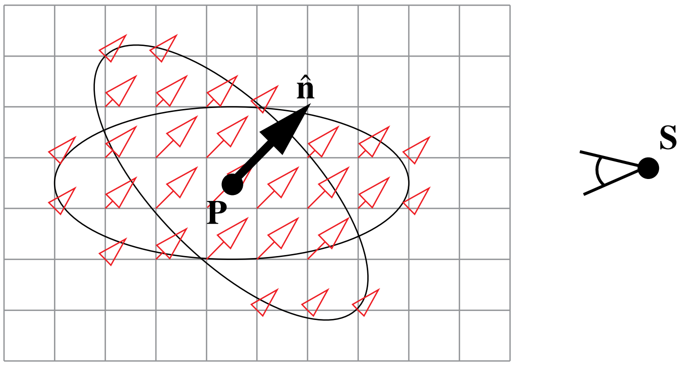


Figure 10: The insertion of a point into the surface probability field. Vectors aligned with the point's surface normal with magnitude depending on the sum of the Point Spread and Sensor Error Model are inserted into the voxel space. **S** is the sensor origin.

the structure of the data and the probability of surface. Instead of storing a scalar probability in each voxel of the occupancy grid, a vector encoding the surface shape and probability is stored. The direction of this vector encodes the consensus surface normal based on all of the local measurements, and the magnitude of the vector encodes the probability of surface. Because a vector is being stored at each voxel instead of a scalar, we call the occupancy grid that holds structured surface information a surface probability field. The magnitude of the surface probability field which conveys the belief in surface existing is termed the surface probability function. A example of the vectors inserted into the surface probability field for a single point at given in Figure 10.

A traditional occupancy grid is incrementally built from measurements by finding the location of a measurement in the occupancy grid and then adjusting the nearby voxels based on the sensor model. The adjustment takes the form of scalar addition of the sensor model probability to the probability that already exist in the voxel. In our occupancy grid representation, when a new measurement is to be inserted into the surface probability field, the voxel containing the measurement is determined. Then for each surrounding voxel, a vector oriented in the direction of the measure ment surface normal with magnitude equal to the sensor model probability at that voxel is calculated. This vector is added to the vector that is already stored in the existing voxel. As shown in Figure 11, using surface normal vectors to combine probabilities has many advantages:

- *Creates consensus surface normal*

  The direction of the vector stored in each voxel is the consensus surface normal because it is the weighted average (based on the sensor model) of the normals from the measurements near the voxel. The consensus surface normal is essential for robust surface extraction.

- *Enforces surface shape coherence*

  Measurements are being added in the surface probability field as vectors. Voxels that are updated with normals pointing in a similar direction will have a larger vector magnitude that voxels that get updated with surface normals that point in different directions. Therefore, only voxels that get updated with measurements of similar surface normal will have high probability of being part of the surface.

- *Prevents mixing of opposing surfaces*

  Opposing surfaces will have opposite surface normals, so between them a gap of low surface probability (due to the cancellation of surface normals) will be generated. This will prevent opposing surfaces from being joined during surface extraction and will aid in the correct generation of blended texture on the extracted surface.

Only voxels, that are close to a sensed point will be updated. For typical scenes, only a small fraction of the voxels that in the volume surrounding the scene will be updated because the scene surface is a small fraction of the volume. Instead of allocating all of the voxels surrounding the scene, we dynamically allocate the voxels when they are needed. Dynamic allocation requires a special voxel storage data structure, called a voxel space, to efficiently determine if a voxel has been allocated. Given the bounding box of the scene points and the size of the voxels, a unique integer key can be assigned to each voxel using the order of the voxels in the raster scan of the bounding box. Allocated voxels are stored in a dictionary (a binary tree that is sorted based on a the keys of its elements) to ensure efficient lookup of already allocated voxels and insertion of newly created voxels. In summary, a voxel space stores the voxel size, volume bounding box and the allocated voxels stored in a dictionary for efficient access.

Given the sensor model and a way of storing voxels, the algorithm for creation of the surface probability field is straightforward. For each scene point determine the voxel that it falls in. Loop over a cube of voxel ids surrounding the location of the point. Search for each voxel id in the voxel space, if the voxel corresponding to the voxel id does not exist, allocate it and insert it into the voxel space. For each voxel in the cube, update its consensus surface normal based on the position of the voxel and the sensor model for the current point (each point has position, surface normal, and

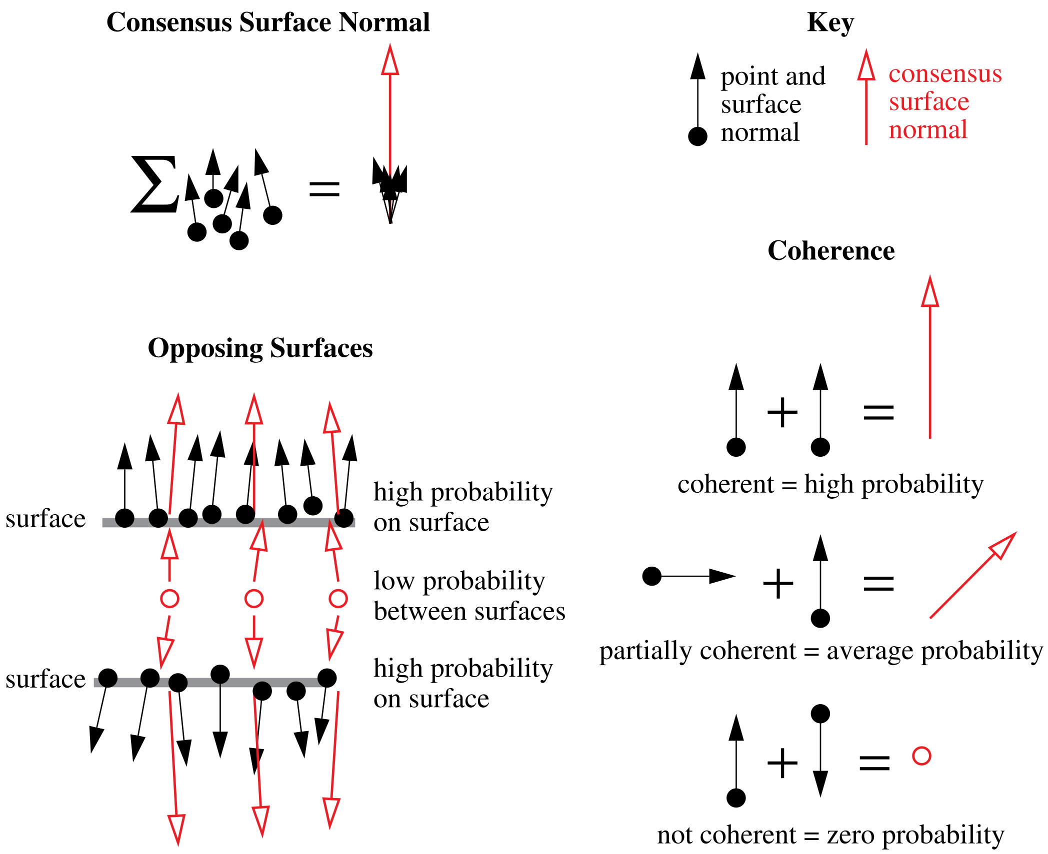


Figure 11: Consensus surface normal definitions. The consensus surface normal is the weighted sum of the normals of surrounding points (top left). Adding probabilities as vectors prevents opposing surfaces from mixing (lower left). Coherence of normals determines magnitude of consensus surface normal (lower right).

```
CreateSurfaceProbabilityField(scene_points S, bounding_box B, voxel_size S)
   voxel_space VS = InitializeVoxelSpace(B,S)
   For each scene_point P in S {
      For each voxel_id I in cube around P {
         if V = LookupVoxel(I,VS) does not exist {
            V = AllocateVoxel(I,VS)
         }
         vector N = SensorModel(P,V)*Normal(P)
         ConsensusNormal(V) = ConsensusNormal(V) + N
         UpdateVoxel(V,VS)
      }
   }
}
```

Figure 12: Pseudo-code of algorithm to create surface probability field.

sensor position information). The end result is the surface probability field for all of the currently sensed scene points where consensus surface normal magnitude corresponds to surface probability. A pseudo-code version of the algorithm is given in Figure 9.

To test our integration method we created six synthetic data sets from the synthetic room model described in the registration section of the paper. The data sets are shown in Figure 13 after registration by the Color ICP algorithm. The location of the sensor origins of the six data sets are shown as shaded spheres that follow a path from one room to the other through the doorway between the rooms. Since multiple data sets taken from different positions throughout the scene, most of the surfaces in the scene are sensed. By creating these synthetic data sets we are able to test our integration algorithm in a controlled setting independently of the scene recovery method. The middle horizontal slice through the final surface probability field showing the magnitude of the consensus surface normal for the 6 synthetic points sets is given in Figure 13.

3.5 Extracting surface from probability field

The surface probability field describes the probability of surface in the volume from which the exact surface can be extracted. Given the scene data, the best estimate of the surface is the set of

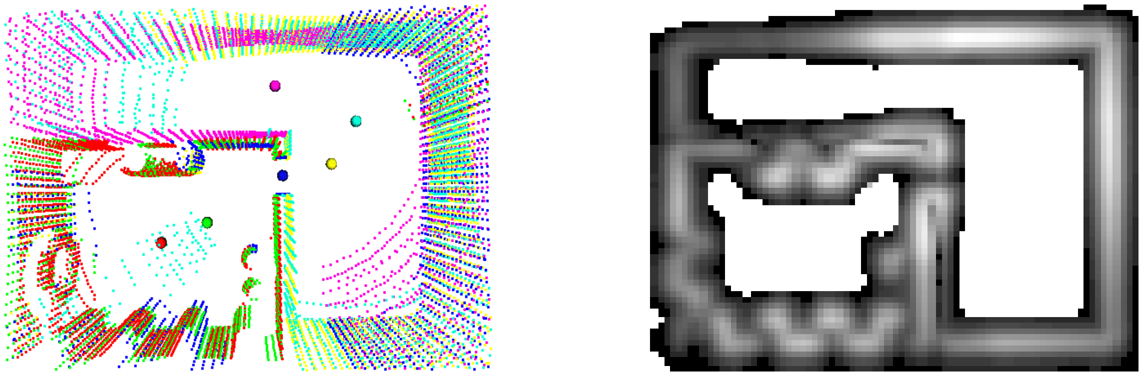


Figure 13: Registered point sets with sensor origins shown as shaded spheres and the middle horizontal slice of surface probability through the voxel space for those points. Notice that only allocated voxels are shown.

ridges in the surface probability function. If an implicit function can be generated that is positive for voxels on one side of the surface and negative for voxels on the other side of the surface, then the exact surface can be extracted from the voxels using an implicit surface polygonizer, such as the Marching Cubes algorithm. The most difficult component of this approach is detecting the ridges in the surface probability function.

In the overview of ridge detection operators by Eberly *et al.* [Eberly *et al.*, 1994], they define a ridge as a point that it a local maximum of some function along a special direction. In image analysis the special directions are usually computed as the eigenvectors of the local Hessian of the function. The Hessian is computed from second order derivatives on the image, hence, in the presence of noise, it cannot be computed robustly. However, if the special direction along which to calculate the local maximum is available through some other robust method, then ridge detection operation will be robust.

In our case, the ridge corresponds to local maxima of the surface probability function in a direction perpendicular to the surface. The consensus surface normal is the direction perpendicular to the surface at each voxel computed as the weighted average of the normals of nearby points. Therefore, the ridge can be computed as the local maxima of the surface probability function in the direction of the consensus surface normal. The direction along which to compute the local maxima does not depend on calculation of second order derivatives, but on the smoothly varying consensus surface normal. Therefore, our method for computing the ridge is more robust than those that use

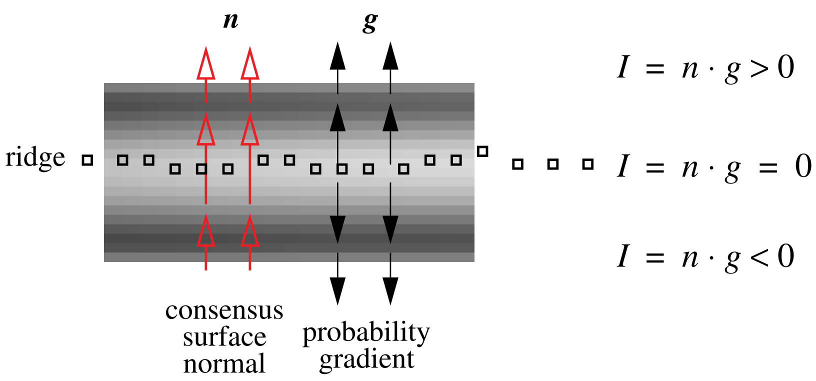


Figure 14: The dot product of the consensus surface normal and the surface probability gradient create an implicit surface function.

traditional ridge operators. Through an intelligent use of structure information available from the scene points, ridge detection is made robust.

Since the exact surface will lie between voxels, an *implicit surface function* is computed at each voxel followed by the Marching Cubes algorithm to polygonize the implicit surface. The implicit surface function is computed as follows. First, the gradient $\mathbf{g}$ of the surface probability function is computed at each voxel in the voxel space through finite differences. The value of the implicit surface function I is then the projection of the gradient onto the consensus surface normal $\mathbf{n}$ at each voxel.

$$I = \mathbf{n} \cdot \mathbf{g} \tag{10}$$

This will give the component of the gradient along the consensus surface normal (the special ridge detection direction), and it will be zero along the ridge. The gradient will always point away from the surface probability ridge while the consensus surface normal will point in a consistent direction on both sides of the surface. Therefore, on one side of the ridge of the surface probability function, the dot product of the gradient with the consensus surface normal will be positive and on the other it will be negative. Figure 14 graphically explains the generation of the implicit surface function.

Valleys in the surface probability function will also cause the implicit surface function to change sign. To prevent this, valleys can be detected by computing the local Hessian H at the point and checking if $\mathbf{n}^{\mathrm{T}} H \mathbf{n} > 0$, where $\mathbf{n}$ is the consensus surface normal. Instead of doing this, we apply a threshold based on surface probability, that will prevent the implicit surface function

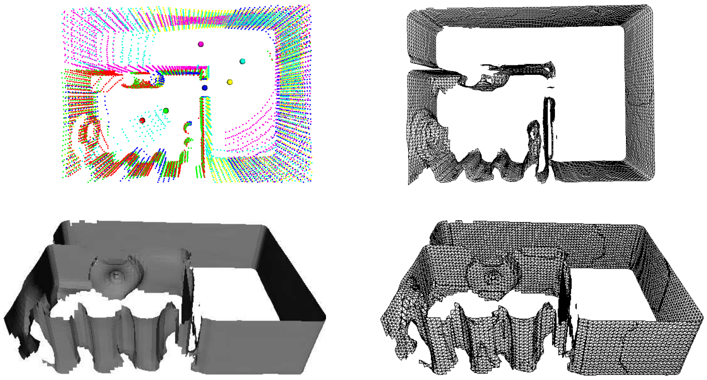


Figure 15: Three views of the consensus surface mesh generated for six registered data sets (merged point distribution shown on the top left). The six small spheres in the point distribution indicate the six camera locations.

from being calculated for voxels with surface probability that is less than a user defined threshold. Since valleys correspond to low probability regions, in most cases, this will prevent the surface from being extracted along valleys. It will also prevent false surfaces from being generated by noisy gradients calculated from small surface probabilities. Using the threshold is a computationally cheaper way to eliminate valleys.

Once the implicit surface function is defined, it is polygonized using the standard Marching Cubes algorithm [Lorensen and Cline, 1987] with a modified lookup table of 22 cases to prevent the creation of holes in the surface [Montani *et al.*, 1994]. The resulting *consensus surface mesh* is the best surface mesh based on the scene data because it corresponds to the surface of highest probability given the sensor model. A vertex of the consensus surface mesh on an edge is determined through linear interpolation of the implicit surface function between voxels connected by the edge.

The consensus surface mesh generated from the 6 synthetic data sets is shown in Figure 15. The walls of the rooms are extracted along with the shape of the objects on the walls. Since the surface probability field is generated through vector addition, two sides of the walls in the middle of the room are kept separated. Some gaps appear where surfaces actually exist in the model because too few points fall on these surfaces for surface extraction.

3.6 Blending texture

Overlap of textured 3-D data sets causes the textures from multiple images to be projected onto a single point in the consensus surface mesh. Texture blending is an informed way of deciding how separate textures are mapped onto the consensus surface mesh. The basic idea is to create a weight for every textured 3-D data set in each voxel. The texture applied to the faces between the voxels is then a weighted average of the texture from the 3-D data sets. A weight for each textured 3-D data set in each voxel can be created from the surface probability field with some extra book-keeping. Each textured 3-D data set adds to the consensus surface normal at a voxel, when a point from the set is near the voxel. The contribution of each data set can be kept track of if the contribution of each data set to the consensus normal is stored as a separate vector in each voxel.

Suppose there are N textured 3-D data sets to be integrated. After all of the points have been inserted into the surface probability field, there are N vectors $\mathbf{n}_i$ in each voxel which measure the contribution of each data set i to the consensus surface normal $\mathbf{n}_c$ of the voxel. The *texture weight* w_i of each data set is then the dot product of the consensus surface normal with the contribution of that data set to the voxel

$$w_i = \max(0, \mathbf{n}_i \cdot \mathbf{n}_c) \tag{11}$$

If $\mathbf{n}_i \cdot \mathbf{n}_c$ is negative, then $\mathbf{n}_i$ is pointing away from the consensus surface normal. This means that the sensor origin of data set i is on the opposite side of the consensus surface, and so data set i should not be contributing texture to the surface. Therefore, if $\mathbf{n}_i \cdot \mathbf{n}_c$ is negative, w_i is set to zero. Using the dot product to create texture weights is the same as setting the texture weight equal to the ratio of area visible to the sensor to actual surface area. This is a reasonable heuristic for vision sensors because as the ratio of visible to actual surface decreases, the reliability of the appearance measured decreases.

Given texture weights for each data set, the texture mapped onto the faces of the consensus surface mesh can be determined. Because the consensus surface mesh was created using the Marching Cubes algorithm, each face in the surface mesh lies in a cube formed by eight voxels. A simple method for determining the texture weights at a point $\mathbf{p}$ on a face in the cube is to trilinearly interpolate the texture weights from the eight voxels based on the 3-D location of $\mathbf{p}$ in the cube. Then the color at $\mathbf{p}$ is the texture weighted average of the color projected onto $\mathbf{p}$ from each data sets. Since trilinear interpolation of image weights is used, the texture will vary continuously over the faces.

A modification must be made to the texture blending to prevent mixing of textures from opposite surfaces. Suppose a face f is being texture mapped. If f is close to an opposite surface (i.e. the other side of a wall) voxels on the opposite side of f can have positive texture weights for data sets from which f is not visible. Therefore, only the texture from data sets that contribute to voxels on the side of f that the consensus surface normal points away from (the sensor side) should contribute to the texture applied to f. The modification to the texture blending is then to only use the texture weights that come from the voxels on the sensor side of f when trilinearly interpolating the texture weights for a point on f. Fortunately, because of the continuity of the consensus surface mesh, the texture weights and hence texture will still be continuously interpolated over the surface mesh.

To apply texture to the consensus surface mesh, a small, square texture map, called a texture cell, is made for each face. The texture cell lies in the plane of the face with the top of the texture cell aligned with the longest edge of the face. The vertices of the longest edge face are mapped to the upper corners of the texture map, so the location of the third vertex in the texture cell can be calculated by geometry. The color of the pixels of the texture cell are then determined by finding the 3-D position of the pixels on the plane of the face followed by texture blending at the point. There is one texture cell for each face, so the texture cells are kept small (i.e., between 8 and 64 pixels wide). Figure 16 shows the geometry for texture mapping onto a face.

3.7 Results

Figure 17 shows the result of texture blending on the surfaces of the consensus surface mesh shown in Figure 15. No misregistration of texture is visible. A close-up of the texture blended surface confirms that there is no misregistration of texture; the pillars and dragon head are clearly visible and distinct with little blurring of boundaries. The integrated texture mapped model coveys the shape of the rooms and their appearance using 14318 faces each with a 16×16 texture cell. This result demonstrates that our shape and appearance registration and appearance algorithms work independently of the shape recovery algorithm.

The next test of our algorithm is the integration of 5 textured 3-D data sets made by performing omnidirectional stereo on the synthetic room model. Camera images are captured from the room model and then composited together into panoramic images. Omnidirectional stereo is then applied to these panoramic images to obtain the resulting textured 3-D data sets. These data sets

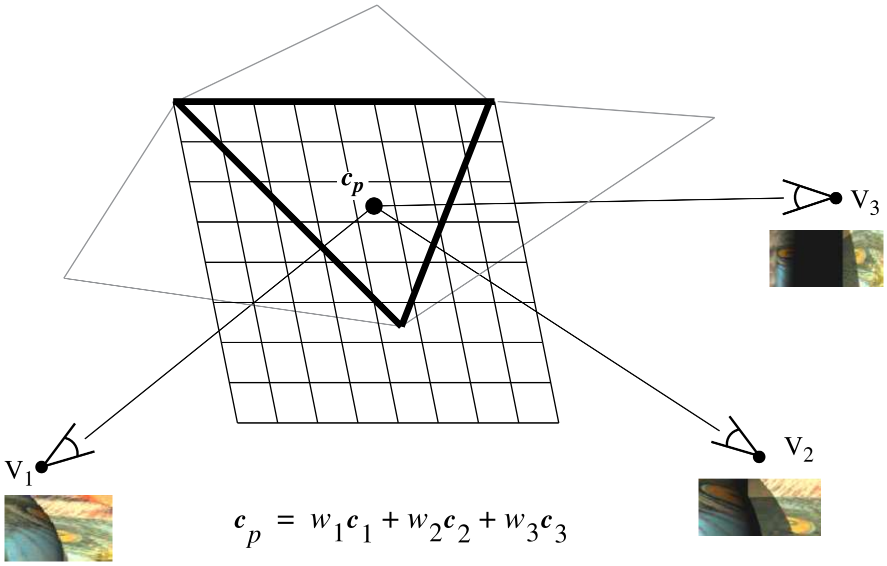


Figure 16: The geometry for creating a texture map cell for a face. The color of each pixel of the cell is the weighted average of the colors projected onto it by data sets that view the pixel. Each face in the consensus surface mesh has an associated texture cell.

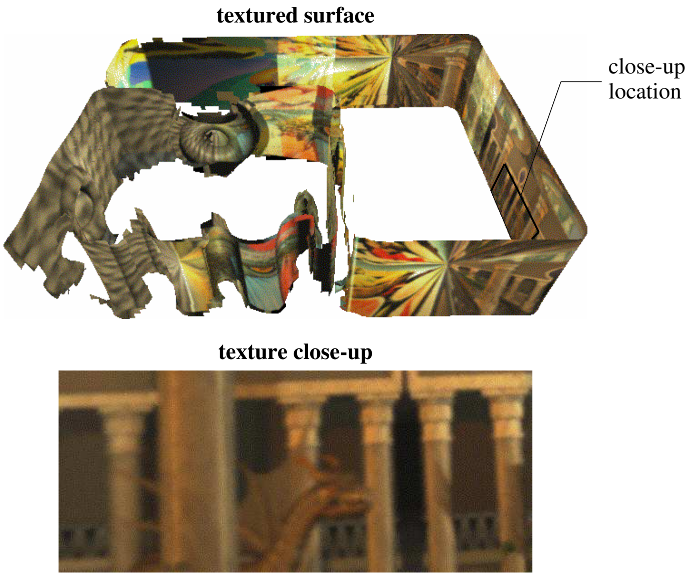


Figure 17: The result of integrating six textured 3-D data sets created directly from a synthetic room model. The complete room model with texture blended on the surfaces of the room is shown as well as a close up of the texture blending.

allow testing of our shape recovery, registration and integration algorithms with perfect synthetic images. The resulting integrated model can also be compared to the synthetic model to analyze the performance of the algorithms with known ground truth. Figure 18 shows the results of integrating five textured 3-D data sets generated using omnidirectional stereo on the synthetic room model into a single model comprising 12990 faces and associated 16×16 texture cells. The registered points, wireframe consensus surface, shaded consensus surface and the texture mapped surface are all shown. The consensus surface clearly shows the shape of the room and the shape of objects on the on the walls of the rooms even when the data is very noisy. In particular the torus on the middle wall of the rooms is clearly defined even though the data defining the torus is corrupted. The integration of texture does not work as well as in the case of previous case of perfect shape recovery. Future work will look into the problem of robust blending texture.

The final demonstration of our algorithms is using real data. Two textured 3-D data sets of an office are created using omnidirectional stereo. The registered points, consensus surface, integrated surface and two close ups of the texture blending are shown in Figure 19. The rectangular shape of the office is reconstructed well enough to blend the texture on its surfaces made from 1099 faces. The closes ups of the texture blending demonstrate two forms of texture blending. One close-up shows the linear blending of texture that was discussed in the previous section. Because of slight misregistration of the 3-D data sets some blurring of texture is visible. Another form of blending uses a maximum operator instead of linear blending. In max blending, the texture of a pixel on a face is taken from a single 3-D data set, the data set with the maximum texture weight at that pixel. This form of blending produces clearer texture, but also allows discontinuities in texture when adjacent pixels take texture from different textured 3-D data sets. Future work will investigate ways to blend texture that combine max blending and linear blending to create continuous and clear texture on the surface of the reconstructed model.

Figures 21 and 22 show the results for merging another two real data sets, this time from two views of the CRL vision lab. These two data sets mostly intersect, except that the first data set includes the small back room in the lab while the other data set does not. The reference panoramic images corresponding to the two data sets are shown in Figure 20.

A difficulty with these data sets stems from the door to the back room (with the stack of VCR's) is relatively narrow, causing the algorithm that creates the 3-D mesh to connect across the doorway for the second data set, as shown at the bottom left of Figure 21. As a result, a preprocessing step of culling points that violates visibility of other data points is performed; the results of this

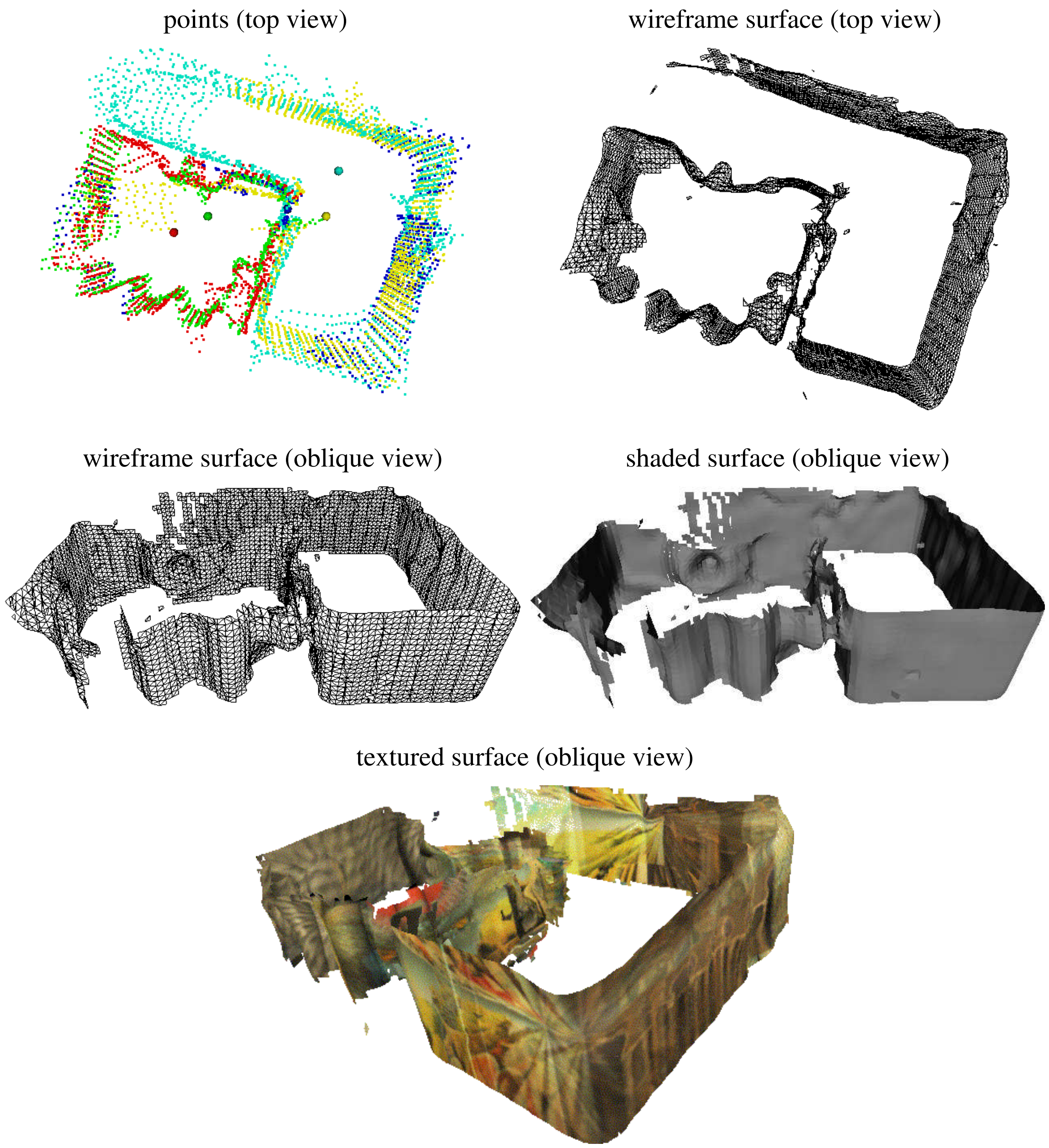


Figure 18: The result of integrating five textured 3-D data sets created from omnidirectional stereo applied to panoramic images created from a synthetic room model. The registered points, wireframe consensus surface, shaded consensus surface and texture mapped surface are shown. Note: the small spheres in the top left figure represent the different camera center locations.

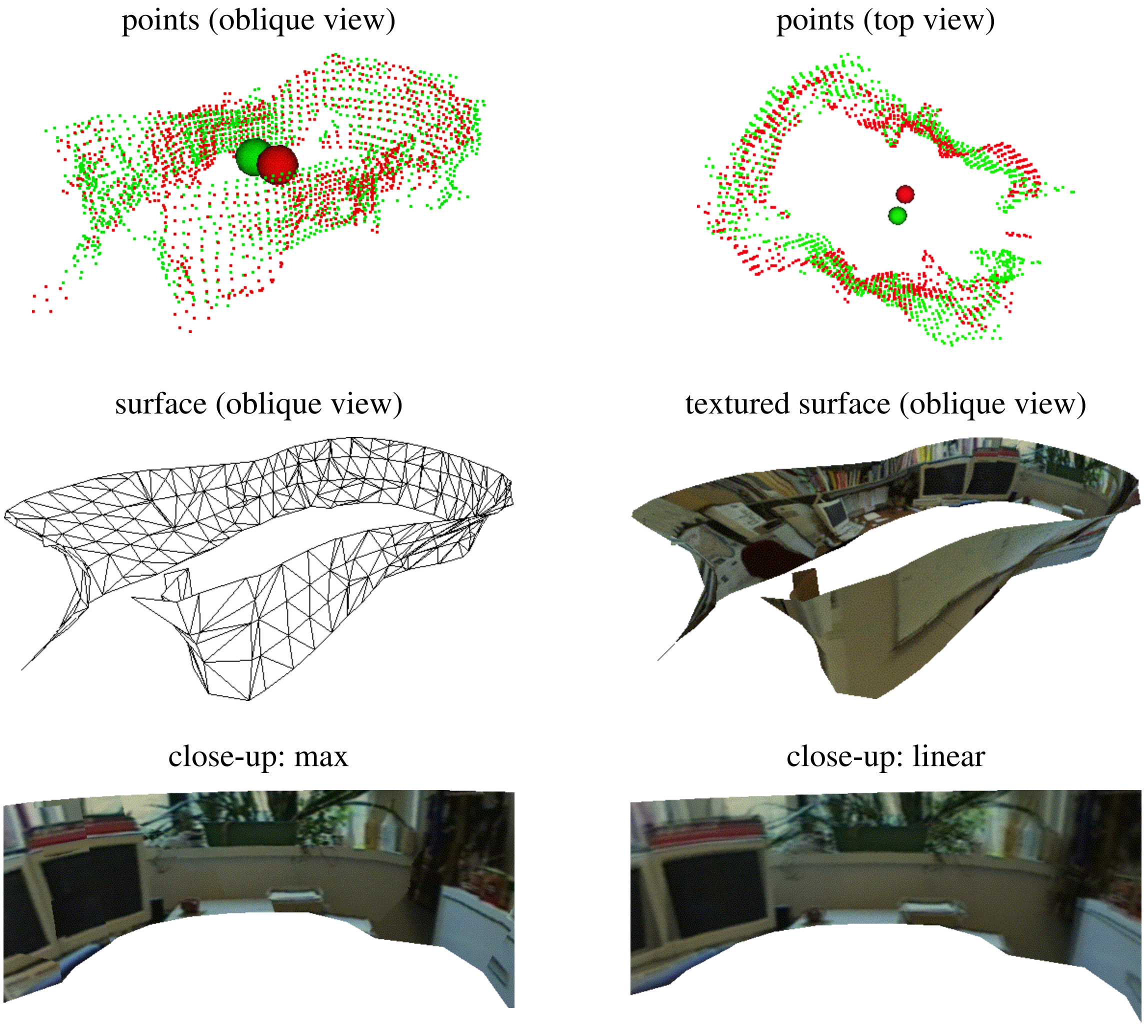


Figure 19: The result of integrating two textured 3-D data sets created with omnidirectional multi-baseline stereo of an office. The registered points, wire frame surface, texture mapped surface and two close-ups of the texture mapping using different blending functions are shown. Max texture blending results in clear texture with more visible discontinuities while linear blending of texture produces less clear texture but with less visible discontinuities. Note: the two small spheres in the top two figures represent the different camera center locations.

Figure 20: Two representative panoramas of the vision lab.

step is shown in Figure 21. The results of merging the two data sets are shown in Figure 22. The discontinuity in the resulting combined texture bears testimony to the recovered shapes at the two different sites not being exact.

4 Implementation

The code for our model merging work is written in C++ and uses LEDA (Library of Efficient Data types and Algorithms) [Naher and Uhrig]. LEDA is a library of data types and algorithms that includes, among others, graph data structures and algorithms to manipulate them. Each vertex, edge, and face of a 3-D scene model has its own data structure, while the connectivity information between the vertices is encoded in a graph. This graph represents the geometrical surface mesh of the 3-D model. Meanwhile, the occupancy grid is represented as a dynamically allocated list of voxel structures; each voxel structure contains the surface normal and probability information. Access to the voxel structure is efficient as it is implemented as a dictionary. The 3-D data merging and modeling program is compiled and run on a DEC Unix Alpha workstation.

While we have written our own version of a 3-D model viewer, we also provide a facility to output our 3-D models as VRML[2] files, primarily because of the increasing popularity of VRML. To view these VRML files, we use a browser called VRweb[3]. Our choice of the VRML browser is strongly influenced by two factors: the complete source code is free, and it can be compiled and run on Unix Alpha workstations.

[2] Virtual Reality Modeling Language, a 3-D version of HTML that enables 3-D models to be directly accessible through the web. The primary web site for VRML is http://www.sdsc.edu/vrml/.

[3] The web site for the VRweb browser is http://www.iicm.edu/vrweb.

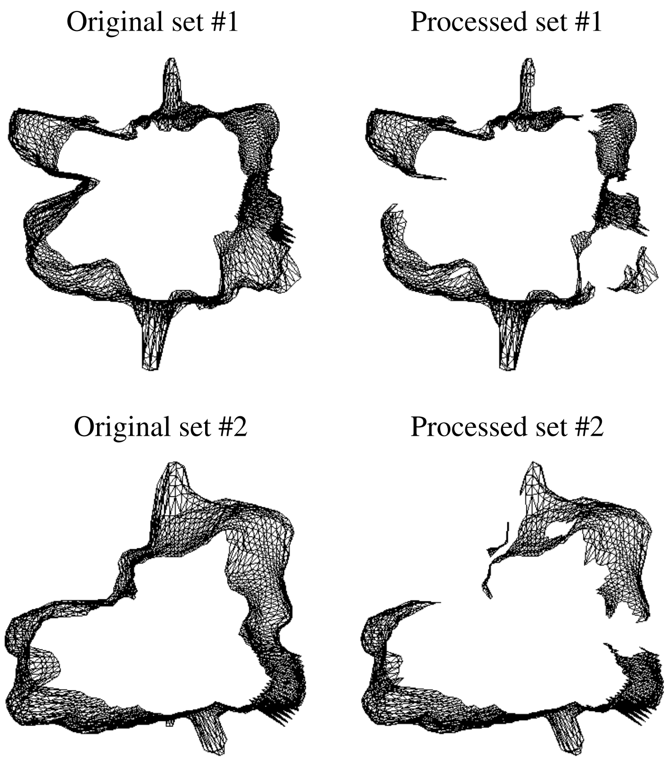


Figure 21: The result of culling data of each data set to ensure non-violation of visibility of other data sets. Left: original (noisy) data sets; right: processed data sets.

5 Discussion and future work

It is not surprising that adding color information to the registration step improves performance. There is a danger, on the other hand, of adding many more local minima with color. This is clearly a function of both the shape and texture distribution. Repetitive shape and texture would have an adverse influence. A solution to this may be to add a simulated annealing-like characteristic to the algorithm to break out of local minima.

One of the problems associated with the integration step is the sensitivity of the results of texture blending to the accuracy of the recovered shape. There is very little recourse to bad input data, though a more sophisticated structure from motion algorithm may be bootstrapped to the registration step to improve both relative camera pose and 3-D data.

The work described here is used to recover 3-D models of indoor scenes for the on-going

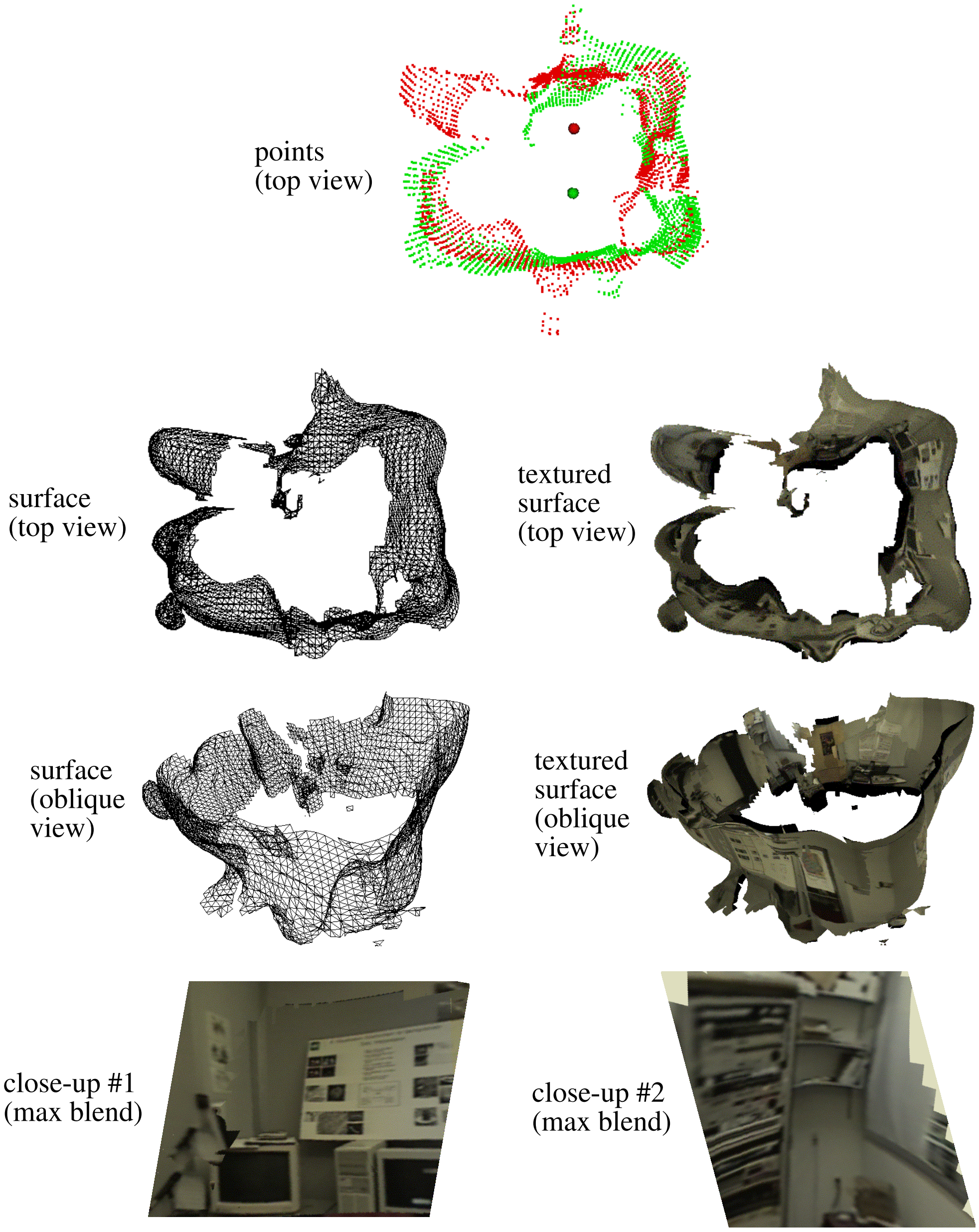


Figure 22: The result of integrating two textured 3-D data sets created with omnidirectional multi-baseline stereo of a lab. The texture of the merged modeled is created using the max texture blending scheme. Note: the two small spheres in the top two figures represent the different camera center locations.

Smart Kiosk project at Cambridge Research Lab, Digital Equipment Corp. [Waters *et al.*, 1996]. The Smart Kiosk can be considered as an enhanced version of the Automatic Teller Machine, with the added capability of being able to interact with the user through body tracking, and gesture and speech recognition. The recovered model of the environment would allow the kiosk to situate the user relative to the environment. As a result, it would enable a more engaging level of user-kiosk interaction, specifically being able to provide relative directions as well as give a virtual tour of the environment. The incorporation of the enhanced feature (using the recovered model of the environment) to the Smark Kiosk is currently underway.

It is perhaps worthwhile to investigate an alternative, view interpolation-based means of generating synthetic views of the model. Recent impressive progress has been made in this area, e.g., [Gortler *et al.*, 1996; Levoy and Hanrahan, 1996; McMillan and Bishop, 1995; Seitz and Dyer, 1996]. However, these methods are not appropriate whenever 3-D structural information of the scene is desired or when certain kinds of views (such as flythroughs involving camera positions very different than those of the known sampled views) are desired.

6 Summary

We have described our approach to merging multiple *textured 3-D data sets*. In our work, the 3-D data sets are recovered using omnidirectional multibaseline stereo, which involves multiple panoramic images of the scene.

Data merging is a two-step process, namely registration and integration. In registering multiple data sets using a variant of the ICP algorithm called the *color ICP*, we not only consider 3-D point location, but also color information. The color information has been shown to improve the registration significantly, especially if there is ambiguity in using just 3-D information.

Once the multiple data sets have been registered, we then extract the complete model. The construction of the merged model is based on voting through occupancy as well as consistency in surface normal direction. The surface of the merged model is recovered by detecting ridges in the occupancy grid, and subsequently polygonized using the standard Marching Cubes algorithm. The texture on the complete model is determined through trilinear interpolation of the overlapping textures corresponding to the original data sets.

References

[Azarbayejani and Pentland, 1995] A. Azarbayejani and A. P. Pentland. Recursive estimation of motion, structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562–575, June 1995.

[Bajaj *et al.*, 1995] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3-D scans. *Computer Graphics (SIGGRAPH'95)*, :109–118, August 1995.

[Besl and McKay, 1992] P. Besl and N. McKay. A method of registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2):239–256, 1992.

[Boissonnat, 1984] J. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Transactions on Graphics*, 3(4):266–286, 1984.

[Chen and Medioni, 1994] Y. Chen and G. Medioni. Surface description of complex objects from range images. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 153–158, 1994.

[Curless and Levoy, 1996] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Computer Graphics (SIGGRAPH'96)*, :303–312, August 1996.

[Eberly *et al.*, 1994] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach. Ridges for image analysis. *The Journal of Mathematical Imaging and Vision*, 4(4):353–373, December 1994.

[Elfes, 1987] A. Elfes. Sonar-based real world mapping and navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):249–265, 1987.

[Faugeras and Hebert, 1986] O. Faugeras and M. Hebert. The representation, recognition and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[Foley *et al.*, 1990] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, Reading, MA, 2nd edition, 1990.

[Friedman *et al.*, 1977] J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

[Gortler *et al.*, 1996] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. *Computer Graphics (SIGGRAPH'96)*, :43–54, August 1996.

[Higuchi *et al.*, 1993] K. Higuchi, M. Hebert, and K. Ikeuchi. *Building 3-D models from unregistered range images*. Technical Report CMU-CS-93-214, Carnegie Mellon University, November 1993.

[Hilton *et al.*, 1996] A. Hilton, A. Stoddart, J. Illingworth, and T. Windeatt. Reliable surface reconstruction from multiple range images. In *Fourth European Conference on Computer Vision (ECCV'96)*, pages 117–126, Springer-Verlag, Cambridge, England, April 1996.

[Hoppe *et al.*, 1992] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (SIGGRAPH'92)*, :71–78, July 1992.

[Kang and Szeliski, 1996] S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. In *Proc.s IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 364–370, June 1996.

[Kang *et al.*, 1995] S. B. Kang, A. Johnson, and R. Szeliski. *Extraction of Concise and Realistic 3-D Models from Real Data*. Technical Report 95/7, Digital Equipment Corporation, Cambridge Research Lab, October 1995.

[Kolb, 1994] C. E. Kolb. *Rayshade User's Guide and Reference Manual*. August 1994.

[Levoy and Hanrahan, 1996] M. Levoy and P. Hanrahan. Light field rendering. *Computer Graphics (SIGGRAPH'96)*, :31–42, August 1996.

[Lorensen and Cline, 1987] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH'92)*, :163–169, July 1987.

[Martin and Moravec, 1996] M. Martin and H. Moravec. *Robot Evidence Grids*. Technical Report CMU-RI-TR-96-06, The Robotics Institute, Carnegie Mellon University, March 1996.

[Matthies and Shafer, 1987] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, RA-3(3):239–248, June 1987.

[McMillan and Bishop, 1995] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics (SIGGRAPH'95)*, :39–46, August 1995.

[Montani *et al.*, 1994] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *Visual Computer*, 10:353–355, 1994.

[Naher and Uhrig] S. Naher and C. Uhrig. *The LEDA User Manual Version R 3.3.1*.

[Rutishauser *et al.*, 1994] M. Rutishauser, M. Stricker, and M. Trobina. Merging range images of arbitrarily shaped objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 573–580, IEEE Computer Society, Seattle, Washington, June 1994.

[Seitz and Dyer, 1996] S. M. Seitz and C. R. Dyer. View morphing. *Computer Graphics (SIGGRAPH'96)*, :21–30, August 1996.

[Shum *et al.*, 1994] H.-Y. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to object modeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 560–565, IEEE Computer Society, Seattle, Washington, June 1994.

[Shum *et al.*, 1995] H.-Y. Shum, K. Ikeuchi, and R. Reddy. *An integral approach to free-form object modeling*. Technical Report CMU-CS-95-135, Carnegie Mellon University, May 1995.

[Simon, 1996] D. Simon. *Fast and Accurate Shape-Based Registration*. PhD thesis, Carnegie Mellon University, 1996.

[Simon *et al.*, 1994] D. Simon, M. Hebert, and T. Kanade. Real-time 3-D pose estimation using a high-speed range sensor. In *IEEE Int'l Conference on Robotics and Automation*, pages 2235–2241, IEEE Society, May 1994.

[Soucy and Laurendeau, 1992] M. Soucy and D. Laurendeau. Multi-resolution surface modeling from multiple range views. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 348–353, IEEE Computer Society Press, Champaign, Illinois, June 1992.

[Sproull, 1991] R. Sproull. Refinements to nearest neighbor searching in k-dimensional trees. *Algorithmica*, 6:579–589, 1991.

[Szeliski and Kang, 1994] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, March 1994.

[Taylor and Kriegman, 1995] C. J. Taylor and D. J. Kriegman. Struction and motion from line segments in multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, November 1995.

[Turk and Levoy, 1994] G. Turk and M. Levoy. Zippered polygonal meshes from range images.

Computer Graphics (SIGGRAPH'94), :311–318, July 1994.

[Veltkamp, 1991] R. Veltkamp. *2D and 3D object reconstruction with the g-neighborhood graph*. Technical Report CS-R9116, CWI Centre for Mathematics and Computer Science, 1991.

[Waters *et al.*, 1996] K. Waters, J. Rehg, M. Loughlin, S. B. Kang, and D. Terzopoulos. *Visual sensing of humans for active public interfaces*. Cambridge, UK, April 1996.

[Zhang, 1994] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *International Journal of Computer Vision*, 13(2):119–152, 1994.