To appear in Image and Vision Computing, 1998.

Registration and Integration of Textured 3-D Data

Andrew Edie Johnson
The Robotics Institute
Carnegie Mellon University
aej@ri.cmu.edu

Sing Bing Kang
Cambridge Research Laboratory
Digital Equipment Corporation
sbk@crl.dec.com

Abstract

In general, multiple views are required to create a complete 3-D model of an object or of a multi-roomed indoor scene. In this work, we address the problem of merging multiple textured 3-D data sets, each of which corresponds to a different view of a scene or object. There are two steps to the merging process: registration and integration. To register, or align, data sets we use a modified version of the Iterative Closest Point algorithm; our version, which we call color ICP, considers not only 3-D information, but color as well. We show experimentally that the use of color decreases registration error significantly. Once the 3-D data sets have been registered, we 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. In addition to occupancy information, we store surface normal in each voxel of the occupancy grid. Surface normal is used to robustly extract a surface from the occupancy grid; on that surface we blend textures from multiple views.

KEYWORDS: scene modeling, 3-D registration, volumetric integration, omnidirectional stereo, textured 3-D data.

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 kiosks,) 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 receiving particular attention, given the labor-intensive and thus expensive nature of creating

This work was done at the Cambridge Research Laboratory and was supported by Digital Equipment Corporation.

models by hand using a 3-D geometric modeler. The problem with creating models of a large and complex scene is that a single view cannot completely convey the shape of the scene--thus merging of multiple views acquired at different locations is usually necessary. In general, merging of multiple views is a two step process: first, the views are registered, then they are integrated into a seamless 3-D model.

Real scenes are described by the shapes of objects in the scene as well as by the appearance (color, texture) of these objects. Therefore, for computer models to be realistic, they must convey both shape and appearance. To meet this end, we have developed a multi-view merging algorithm that integrates the shape *and* appearance of complex scenes. Like many multi-view merging algorithms, our approach has two components: a registration stage that aligns multiple views of a scene and an integration stage that combines the multiple views into a single seamless model. For registration, we use a modification of the Iterative Closest Point algorithm [2][4][30] that aligns two surfaces using color and shape information. For integration, we have developed a volumetric range image integration algorithm [6][12][13][29], based on occupancy grids, that can be used to merge shape and appearance from multiple textured 3-D data sets.

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 [15]. 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

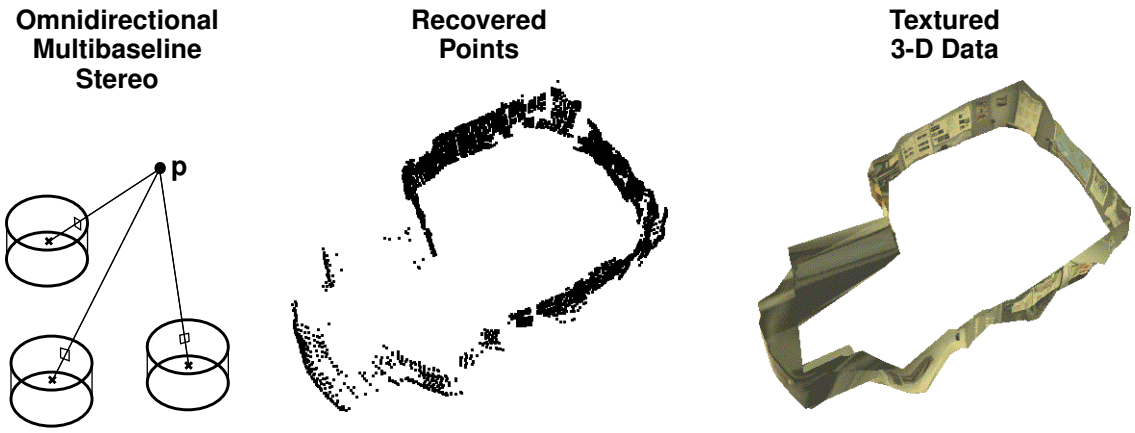


Figure 1. Shape recovery using omnidirectional multibaseline stereo.

summarized in Figure 1. We provide only a brief outline of the approach here; full details can be found in [15]. The approach is straightforward: at each camera location in the scene, sequences of images are captured while the camera is rotated about the vertical axis passing through the camera optical center. Each set of images is then composited to produce panoramas at each camera location. The stereo algorithm is then used to extract 3-D data of the scene. A surface represented as a triangular surface mesh is constructed from the 3-D data. Finally, a textured 3-D data set results when the surface mesh is rendered with the texture provided by the reference 2-D input image.

Given multiple textured data sets recovered using omnidirectional multibaseline stereo, the first step in the 3-D merging process is data set registration.

3. 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 objects in the scene, the registration algorithm used must be able to handle free-form surfaces. The Iterative Closest Point algorithm (ICP) [2][4][30] 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.

3.1 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 & McKay [2] 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}. ICP is an elegant way to register free-form surfaces because it is intuitive

and simple. Besl & 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 corresponding scene point due to occlusion in the scene.

Zhang [30] 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 [10][25] 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. Zhang showed good results with stereo data, which motivated our use of the ICP algorithm for registration.

Chen and Medioni [4] have developed an interactive range image registration algorithm based on minimizing the normal distance between two surfaces. Their algorithm will converge faster than the ICP algorithms described above because it reduces the number of conflicting constraints imposed by closest point correspondences. However, major modifications to the algorithm would be necessary to make it suitable for registration based on shape and appearance.

We have developed an ICP algorithm that builds on the algorithm presented by Zhang [30]. 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.

3.2 Color ICP

During integration of textured 3-D data, shape as well as texture are integrated to form what is termed 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. 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 p_1 and p_2 with positions $\boldsymbol{x_1} = (x_{11},x_{12},x_{13})$ and $\boldsymbol{x_2} = (x_{21},x_{22},x_{23})$ and colors $\boldsymbol{c_1} = (c_{11},c_{12},c_{13})$ and $\boldsymbol{c_2} = (c_{21},c_{22},c_{23})$. then the 6-D l_2 color/shape distance between the points is

$$d_6(p_1, p_2) = [(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + (x_{13} - x_{23})^2 + \alpha_1(c_{11} - c_{21})^2 + \alpha_2(c_{12} - c_{22})^2 + \alpha_3(c_{13} - c_{23})^2]^{\frac{1}{2}} \quad (1)$$

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. The 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 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

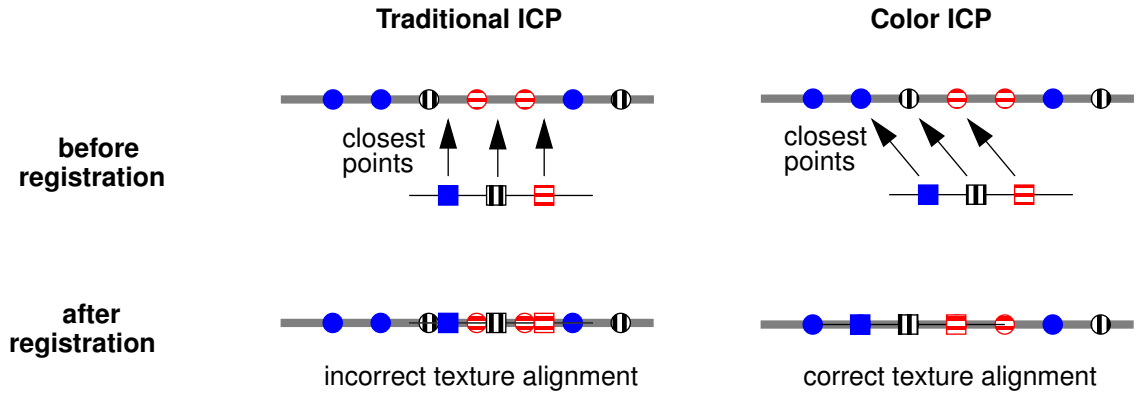


Figure 2. Demonstration of the use of color in registration. In traditional ICP closest points depend only on shape, so it can produce incorrect texture registration. Since closest points depend on color and shape in Color ICP, it will aligns texture correctly.

image to obtain their color. The extra points for each face are created on a regular grid of fixed size attached to each face. To speed up registration, 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. Greater registration accuracy could have been obtained by increasing the resolution of the subsampling grid, but we found it unnecessary. 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 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 (1), 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 commence. For each color-point m_i in M, the algorithm finds the closest color-point s_i in S using the 6-D tree for S. Given the (m_i, s_i) pairs, it then computes the rigid transformation T that will minimize the 3-D euclidean distance between their spatial coordinates using the quaternion method of Faugeras and Hebert[8].

$$T(t_x, t_y, t_z, r_x, r_y, r_z) \;=\; \min_{T} \; \sum_{i} \left\| \boldsymbol{s_i} - T(\boldsymbol{m_i}) \right\|^2 \tag{2}$$

Finally, it transforms the points in M by T and repeats the iterations until the convergence criterion is met.

To make Color ICP robust when the model is not a proper subset of the scene, we have incorporated the dynamic maximum distance threshold employed by Zhang[30]. 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 incremental translation and the magnitude of a vector made from the incremental rotation angles fall below separate thresholds.

$$d_t = \sqrt{t_x^2 + t_y^2 + t_z^2} < H_t \qquad d_r = \sqrt{r_x^2 + r_y^2 + r_z^2} < H_r \tag{3}$$

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 is reduced. A color model that meets this criterion is the YIQ color model [9]. 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 (1) is $(y,i,q) = (c_1,c_2,c_3)$.

The scale of color with respect to 3-D position is determined by the α vector in (1). To reduce the effect of intensity variation 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 $\alpha = (0.1,1.0,1.0)$. Since the spatial resolution of the 3-D textured data sets that we are merging is on order of one unit, this scale factor makes the intrinsic color of points have an effect that is on order of the variation of the 3-D position of the points. By registering two data sets using different values for α, we found that accuracy of the registration is not very dependent on the relative scale between color and shape.

3.3 Results

An example registration is shown in Figure 3. At the top of the figure are shown two synthetic textured 3-D data sets (in wireframe and texture mapped) generated from a room model created using the Ray Shade modeling package [16]. The room has four 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 (mandrill face,...) to add extra texture to

Table 1: Comparison of registration results to ground truth.

registration algorithm	transformation parameters						errors	
	t_x	t_y	t_z	r_x	r_y	r_z	E_t	E_r
shape only	1.993	0.793	-0.173	0.064	1.331	-0.232	0.173	1.353
color and shape	1.994	0.794	-0.010	-0.151	-0.060	0.049	0.013	0.170
color only	1.856	0.655	-0.082	1.019	-0.870	-0.176	0.220	1.826
Correct	2.000	0.800	0.000	0.000	0.000	0.000	0.000	0.000

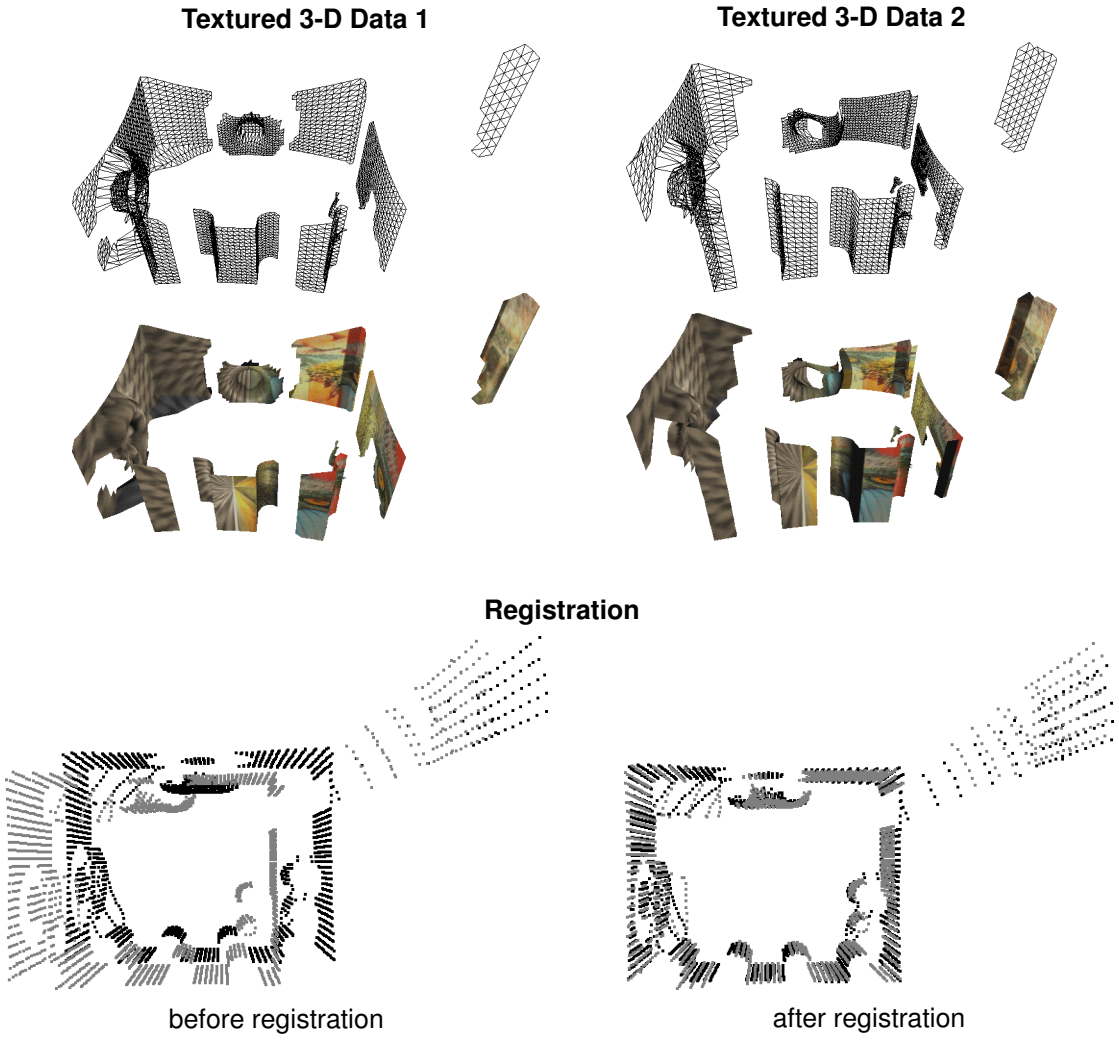


Figure 3. Two textured 3-D data sets of a synthetically generated room shown in wireframe and texture mapped (top). A top view of the points in the sets before and after registration by the Color ICP algorithm (bottom).

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 synthetic data sets is known. At the bottom of Figure 3 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 registration results to ground truth can be made. To demonstrate the benefits of adding color to registration, three experiments were performed using the data sets shown in Figure 3. First, the data sets were registered using ICP based on shape alone; no color information was used in the registration. Next, the data sets were registered using color and shape information. Finally the data sets were registered using only the color information of the points; the 3-D positions of all points were set to zero. Table 1 shows the transformation parameters $(t_x, t_y, t_z, r_x, r_y, r_z)$ calculated for these three different registration experiments and the correct ground truth transformation. Also shown are the least-squares deviation from ground truth in translation (E_t) and rotational (E_r) for the computed transformations. Comparison of the transformation deviations shows that color

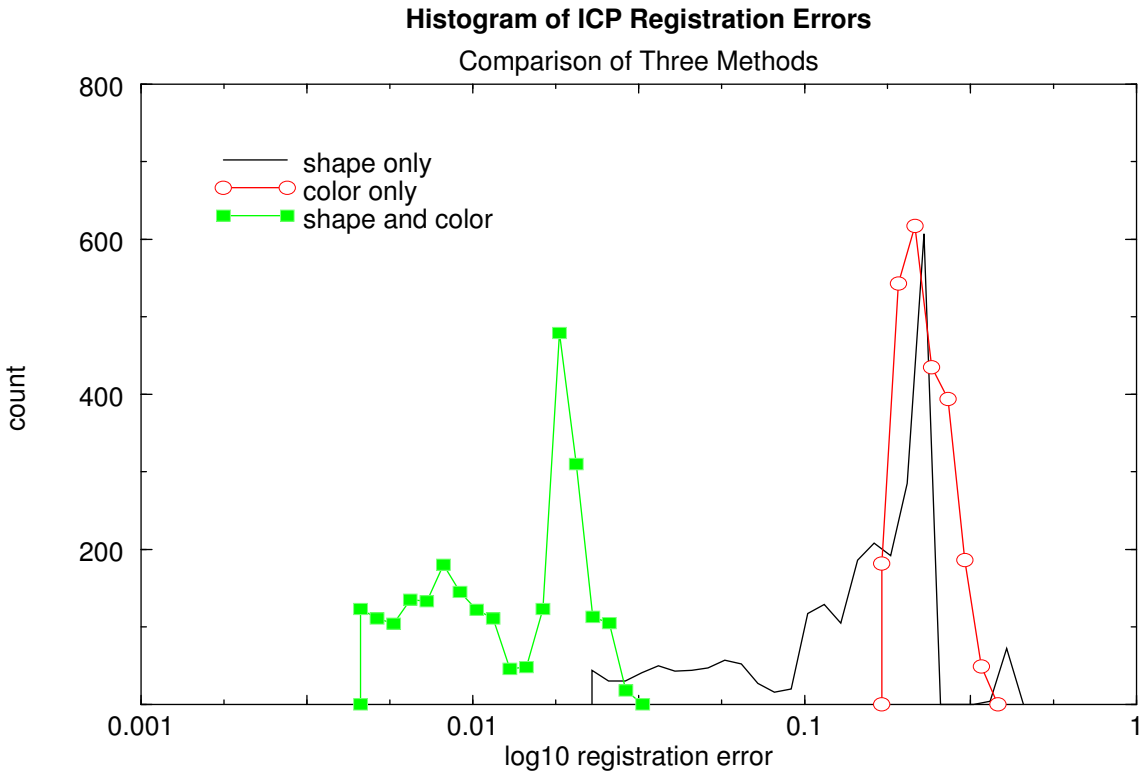


Figure 4. Histogram of registration errors for three versions of ICP: shape only, color only, and shape and color. The histogram clearly shows that the using color and shape improves registration by an order of magnitude over shape alone or color alone.

and shape registration performs significantly better than the registration using shape alone and registration using color alone. As mentioned previously, to integrate texture, the rotational registration error should be on order of tenths of a degree. The registration produced using color and shape is within this 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 the transformed location of the 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 in Figure 4 for the three registration experiments. The median of the errors for the shape only experiment is around 0.10, the median for color only experiment is around 0.20 and the median of the errors for the color and shape algorithm is around 0.01. For this data set, the performance of the Color ICP is an order of magnitude better than the performance of ICP that uses shape alone.

4. Integration

After registration, the data sets are integrated to produce a seamless 3-D textured model. The integration step involves the use of occupancy grids based on sensor modeling and robust ridge detection to recover composite 3-D surfaces. Occupancy grids [7][18] represent the likelihood of 3-D spatial occupancy through voting. This representation is very attractive for integrating 3-D textured data sets because it is incremental, simple, allows for free-form objects, and flexible in allowing the incorporation of data from different sensors and their models.

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 [3] developed an algorithm for efficient computation of the Delaunay tetrahedronization of space. Veltkamp [27] 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. [13] 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. [1] use alpha-shapes and Bernstein-Bezier forms to construct smooth surfaces from a set of unorganized points. Unstructured point sets have no sensing information, so these algorithms cannot make use of a sensor model or viewing direction to improve results in the presence of noisy data. 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 [24] 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 [26] 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. [22] 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 [5] 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 a the Marching Cubes [17] 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. [12], Curless and Levoy [3] and Wheeler [29] 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.

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 [7]. 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 to 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.

4.1 Sensor model

Before accumulating surface evidence in an occupancy grid, a sensor model must be determined. Our sensor model G has two components: the sensor error model and the point spread model. The sensor error model G_E is an approximation of a true stereo error model and is represented as an ellipsoidal gaussian distribution centered at the measured 3-D point whose axis is oriented along the line of sight [19]. The point spread model G_S is used to promote the generation of surfaces from discrete data. It is represented as a cylindrical gaussian, centered at the sensed point, whose axis is aligned with the local surface normal.

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

$$G(\lambda, \sigma_\alpha, \sigma_\beta, \sigma_\gamma, \sigma_\delta) \;=\; \lambda G_E(\sigma_\alpha, \sigma_\beta) + (1-\lambda) G_S(\sigma_\gamma, \sigma_\delta) \tag{4}$$

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

Sensor Error Model GE **Point Spread Model GS**

Figure 5. 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.

processed.

Analytically, the sensor error model G_E has the form of a cylindrically symmetric gaussian with its axis aligned with the local viewing direction

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

where α is the distance of the query point $\boldsymbol{x}$ from the unit viewing vector $\hat{\boldsymbol{v}}$ and β is the distance of the query point $\boldsymbol{x}$ along the unit viewing vector. $\boldsymbol{S}$, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are the world coordinates of the sensor, the data point, and an arbitrary point, respectively. The spread of the gaussian can be characterized by two parameters, ${\sigma_\alpha}^2$ the variance perpendicular to the viewing direction and ${\sigma_\beta}^2$ the variance along the viewing direction. A 2-D slice of the sensor error geometry is shown in Figure 5.

Matthies and Shafer [19] showed 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 can be automatically set by analyzing local changes in distance from the sensor which characterize the noise in the recovery of the 3-D points. Consider a point $\boldsymbol{p}$ from surface mesh M that has N_M points and sensor origin $\boldsymbol{S}$. Call the local surface mesh neighborhood of $\boldsymbol{p}$ (points connected to $\boldsymbol{p}$ by the mesh), L_P with N_p points. The RMS spread in distance d_{rms} for M is calculated as follows:

$$
\begin{gathered}
\bar{d} = \frac{1}{N} \sum_{\boldsymbol{p} \in L_P} \|\boldsymbol{p} - \boldsymbol{S}\| \qquad d_{\boldsymbol{P}} = \frac{1}{N_P} \sum_{\boldsymbol{p} \in L_P} \bar{d} - \|\boldsymbol{p} - \boldsymbol{S}\| \\
d_{rms} = \frac{1}{N_M} \sqrt{\sum_{\boldsymbol{p} \in M} d_P^2}
\end{gathered}
\tag{6}
$$

d_{rms} measures the average local change in distance along the viewing direction 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 on 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 model is added

to the sensor model. The point spread model has the form of a cylindrically symmetric gaussian with its axis aligned with the local surface normal

$$G_S(\gamma, \delta, \sigma_\gamma, \sigma_\delta) = \frac{1}{\sqrt{2\pi}} \sqrt{\frac{1}{\sigma_\gamma^2} + \frac{1}{\sigma_\delta^2}} \exp\left(-\frac{1}{2}\left(\frac{\gamma}{\sigma_\gamma}^2 + \frac{\delta}{\sigma_\delta}^2\right)\right) \tag{7}$$

$$\gamma = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x} - \boldsymbol{x} \cdot \hat{n}} \qquad \delta = \boldsymbol{x} \cdot \hat{n} \qquad \boldsymbol{x} = \boldsymbol{Q} - \boldsymbol{P}$$

where g is the distance of the query point x from the unit surface normal $\hat{\mathbf{n}}$ and d is the distance of the query point $\boldsymbol{x}$ 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 5.

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

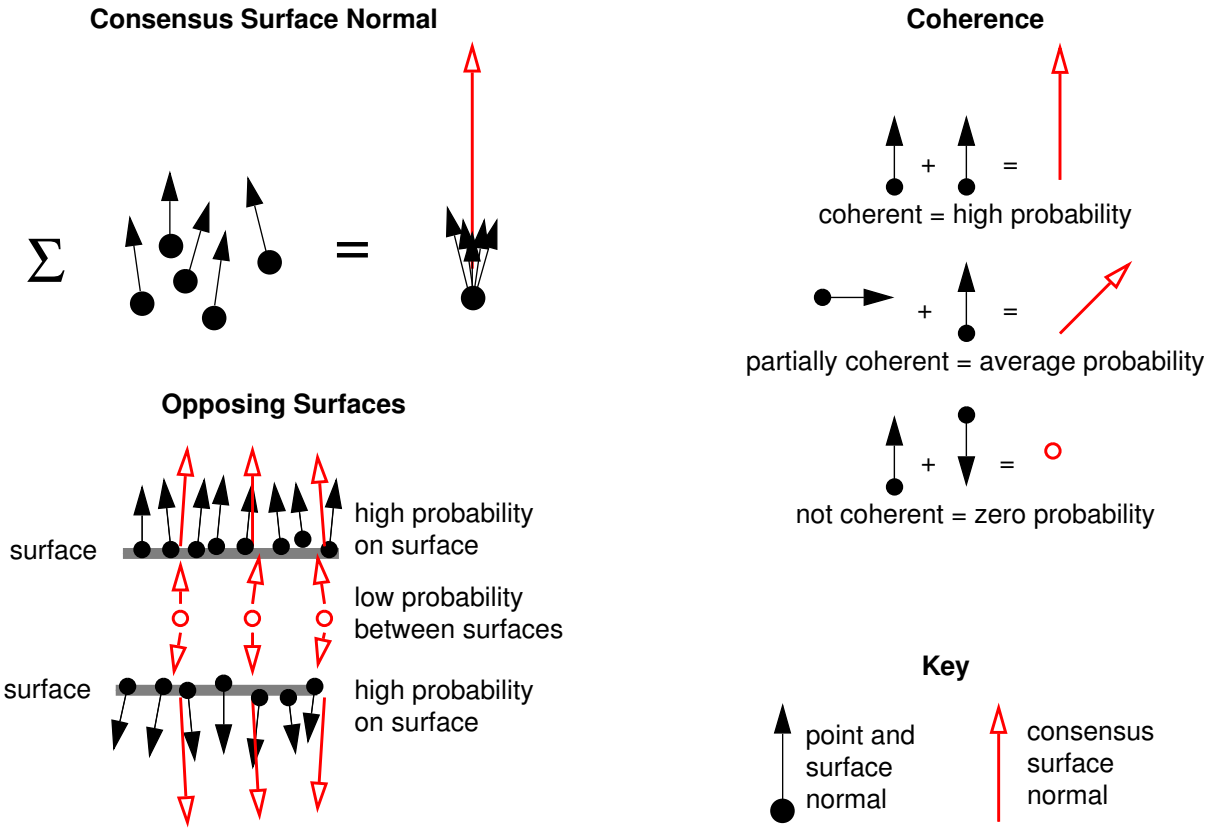


Figure 6. Consensus surface normal definitions. The consensus surface normal is the weighted sum of the normals of surrounding points (left). Adding probabilities as vectors prevents opposing surfaces from mixing (middle). Coherence of normals determines magnitude of consensus surface normal (right).

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 mesh resolution r_{av} for a surface mesh M with N_E edges is

$$r_{av} = \frac{1}{N_E}\left(\sum_{p_i, p_j \in M} \left\| p_i - p_j \right\| \right). \tag{8}$$

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

An example of the recovered surface likelihood functions is shown in Figure 8(b) for six registered data sets. Brighter values correspond to higher probability.

4.2 Recovering consensus surface

While the traditional occupancy grid stores only the likelihood of occupancy, we encode the *consensus surface normal* and surface likelihood in a 3-vector in each voxel. The magnitude of the vector corresponds to surface likelihood and the direction corresponds to the consensus surface normal (likely local surface normal). As shown in Figure 7, vector addition instead of scalar addition is used to update the occupancy grid. When a new sensor measurement is inserted into a voxel, a vector oriented in the direction of the measurement surface normal with magnitude equal to the sensor model at that voxel is added to the vector already stored in the voxel. Therefore, each voxel contains a weighted sum of the surface normals of nearby data points; this gives an estimate of the most likely surface normal at that point in space given the sensed data. Using a vector representation improves the surface generation because it provides the most likely local surface normal, enforces shape coherence and prevents mixing of

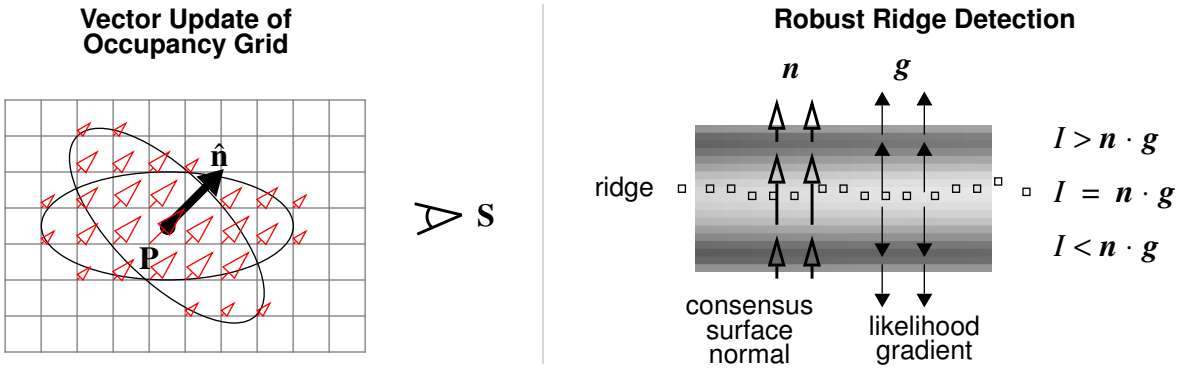


Figure 7. The occupancy grid stores consensus surface normal and is updated by each data point using vector addition (left). The consensus surface normal defines the direction along which to search for local maxima in surface likelihood; this is the key to robust ridge detection (right).

spatially close, but opposing surfaces which is particularly important when integrating texture. The benefits of the vector representation are demonstrated in Figure 6.

The surface of the merged data set is recovered by detecting ridges in surface likelihood. In our case, the ridge can be computed as the local maxima of the surface likelihood in the direction of the consensus surface normal. More specifically, the gradient of surface likelihood is computed using finite differences at each voxel of the occupancy grid. Next, the dot product of the surface likelihood gradient ($\boldsymbol{g}$) and the consensus surface normal ($\boldsymbol{n}$) are computed at each voxel. As shown in Figure 7, this dot product defines an implicit surface function (I) which will be positive on one side of the ridge and negative on the other side.

$$I = \boldsymbol{n} \cdot \boldsymbol{g} \tag{9}$$

This ridge detection method is more robust than traditional ridge operators because the direction along which to calculate the local maxima for ridge detection is already given by the consensus surface normal. Usually ridge detection requires computation of second order surface derivatives to determine this direction--a computation without a robust solution.

The implicit surface function is then polygonized using the standard Marching Cubes algorithm [17] with a

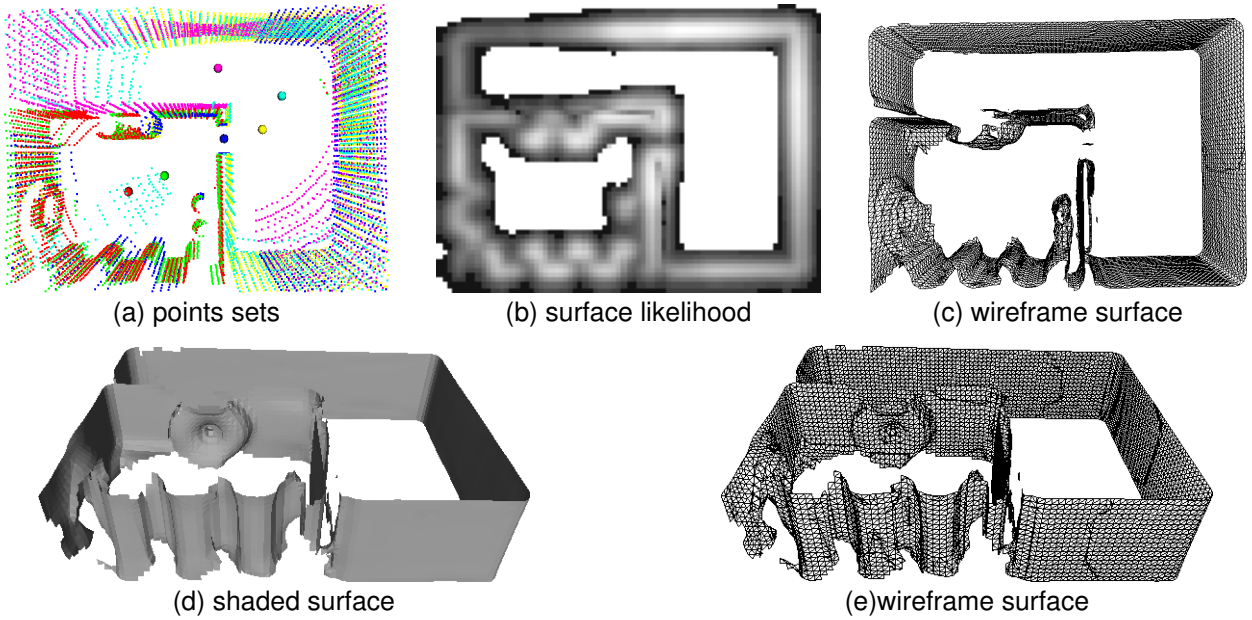


Figure 8. Registered points sets (a) with sensor origins shown as shaded spheres and the middle horizontal slice of surface likelihood through the voxel space for those points (b). Notice that only allocated voxels are shown. Three views of the consensus surface mesh generated for six registered data sets are also shown (c)-(e).

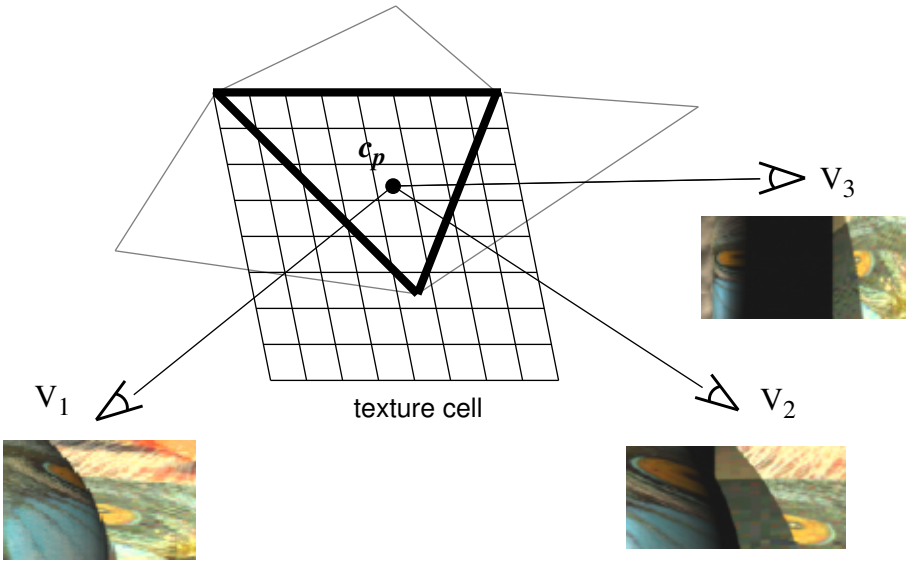
modified lookup table of 22 cases to prevent the creation of holes in the surface [20]. The consensus surface mesh generated from the 6 synthetic data sets is shown in Figure 8. Once the consensus surface has been recovered, the next step is to generate its texture.

4.3 Blending texture

Texture blending is done by weighted averaging of overlapping textures from the original contributing data sets. The *texture weight* is a function of the angle between the surface normal and the viewing direction; textures that subtend more than 90˚ with the consensus surface normal are discarded.

Suppose there are N textured 3-D data sets to be integrated. For texture blending, we store an additional vector in each voxel for each data set being integrated; this vector encodes the vector contribution of the data set to the consensus surface normal of that voxel. Suppose there are N vectors $\boldsymbol{n_i}$ in each voxel, and those vectors measure the contribution of each data set i to the consensus surface normal $\boldsymbol{n_c}$ 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

$$w_i = max(0, \boldsymbol{n_i} \cdot \boldsymbol{n_c}) . \tag{10}$$



$$\boldsymbol{c}_p = w_1 c_1 + w_2 c_2 + w_3 c_3$$

Figure 9. 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.

If w_i is negative then $\boldsymbol{n_i}$ 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, so data set i should not be contributing texture to the surface. Therefore, if w_i is negative, it 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. Furthermore, we are eliminating the need for ray-tracing by storing the relative contribution of each data set in each voxel.

Because the consensus surface mesh was created using the Marching Cubes algorithm, each face in the surface

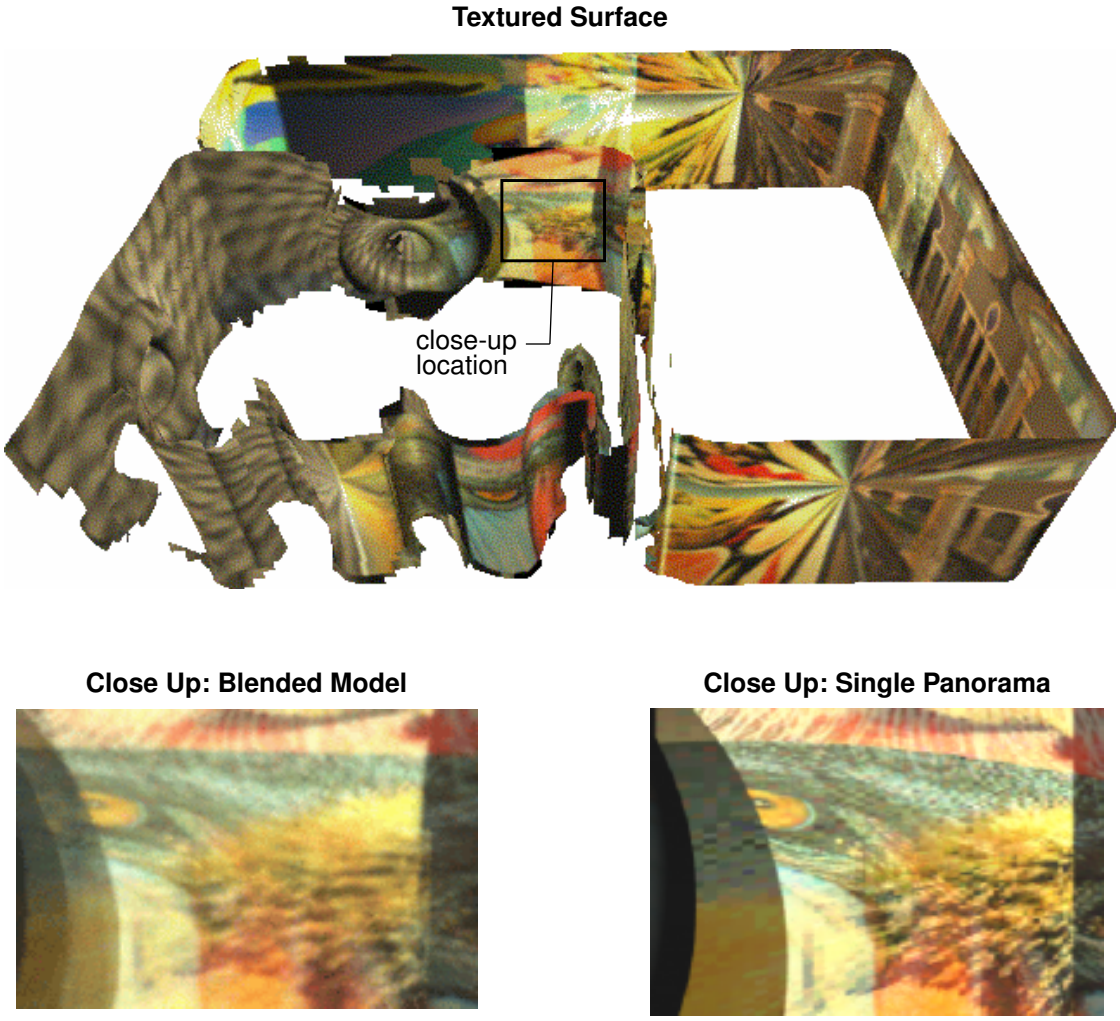


Figure 10. The result of integrating six textured 3-D data sets created directly from a synthetic room model. The complete room model with texture blended on the surfaces of the room is shown as well as a close up of the texture blending. A portion of a single panorama image corresponding to the close up area is shown as well to demonstrate the effect of registration on texture blending.

mesh lies in a cube formed by eight voxels. A simple method for determining the texture weights at a point $\boldsymbol{P}$ on a face in the cube is to trilinearly interpolate the texture weights from the eight voxels based on the 3-D location of $\boldsymbol{P}$ in the cube. Then the color at $\boldsymbol{P}$ is the texture weighted average of the color projected onto $\boldsymbol{P}$ from each data sets. Since trilinear interpolation of image weights is used, the texture will vary continuously over the faces.

To apply texture to the consensus surface mesh, a small, square texture map, called a texture cell, is made for each face. 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 trilinear texture blending at the point. The geometry of texture blending is illustrated in Figure 9. The quality of the appearance of the consensus surface is dependent on the size in pixels of the texture cells used to map appearance onto the surface. We set the size of the texture cells so that, on average, each pixel in a texture cell corresponds to a single pixel from the contributing data sets. This strikes a balance between over-sampling and under-sampling texture from the original data sets. Figure 10 shows the result of texture blending six textured 3-D data sets. To demonstrate the effectiveness of texture blending, a close up of the appearance of the model, with texture from three of the six data sets, is shown as well as a section of a panoramic image that contributes to the close up area.

An alternative method of texture blending is to set the color in each pixel of a color cell to the color of the panorama that has the maximum texture weight for that pixel. In general, using the maximum weight to blend textures will result in crisper textures. However, some discontinuities in texture due to registration errors may appear when the maximum texture weight between adjacent pixels switches from one panorama to another. The trade-offs between smooth texture blending and maximum texture blending are shown clearly in Figure 11. Smooth texture blending results in smooth transitions between texture from different panoramas, but produces some blurring of appearance. Maximum texture blending can be as crisp as the original images, but can result in discontinuities in appearance.

5. Results

Figure 11 shows the results of an experiment where two views of a real office scene are registered and integrated to form a more complete and less noisy model of the room. Although the geometric accuracy of the final model could clearly be improved, this result demonstrates that our algorithms for registration and integration of textured 3-D data sets can be used to combine appearance from multiple views of real scenes even when the input data is sparse and noisy. Two close-ups of the final model show clearly the trade-off between blending textures smoothly and blending textures using the maximum texture weight criterion.

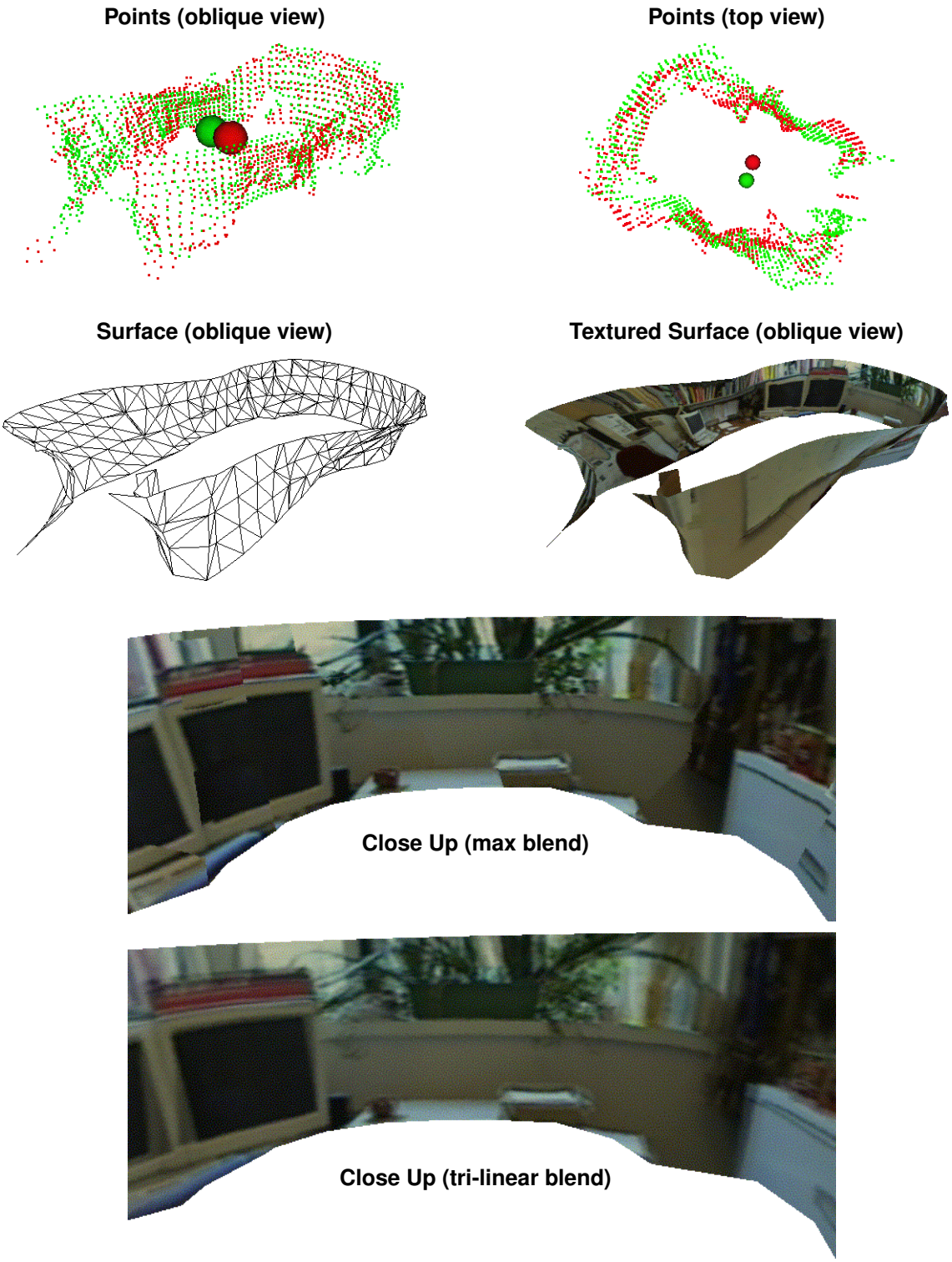


Figure 11. The result of integrating two textured 3-D data sets created with omni directional stereo of an office. The registered points, wire frame surface, texture mapped surface and two close-ups of the texture mapping using different blending functions are shown. Max texture blending results in clear texture with more visible discontinuities while tri-linear blending of texture produces less clear texture but with less visible discontinuities.

Figure 12 and Figure 13 show the results for merging two other real data sets, this time from two views of our 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 12.

A difficulty with these data sets stems from the fact that 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 12. As a result, a preprocessing step that culls points that violate visibility of other data points is performed; the results of this step is shown in Figure 12. The results of merging the two data sets are shown in Figure 13. The discontinuity in the resulting combined texture bears testimony to the fact that the recovered shapes at the two different sites are not exact.

6. Implementation

The code for our model merging work is written in C++ and uses the Library of Efficient Data types and Algorithms (LEDA) [21]. 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

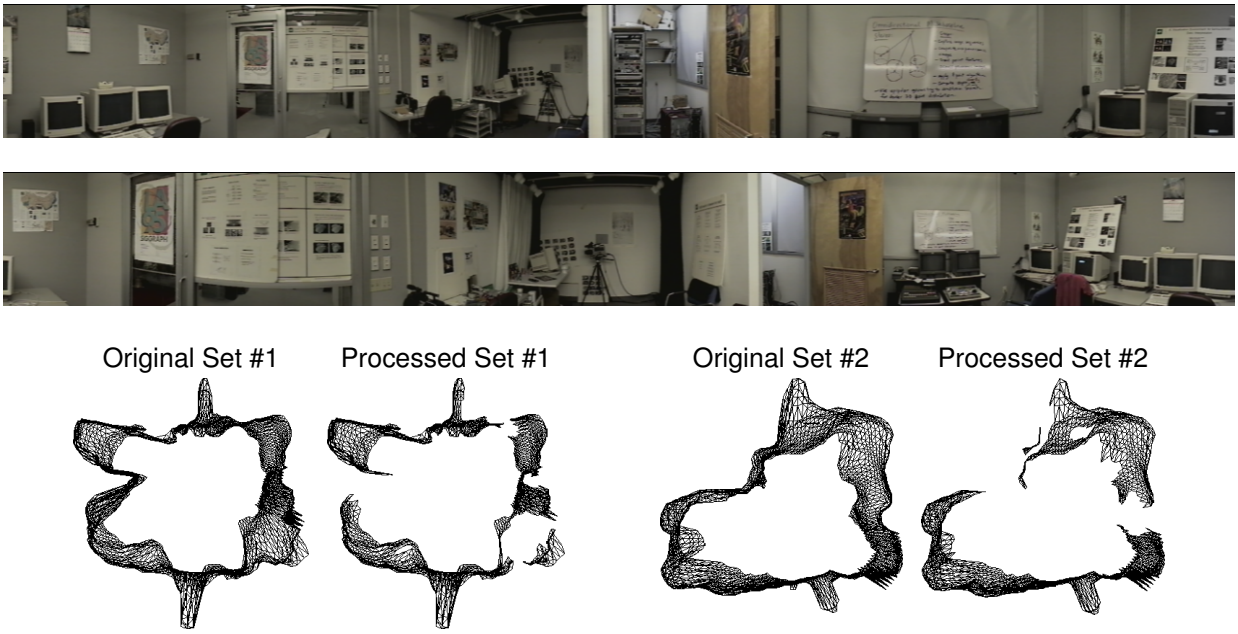
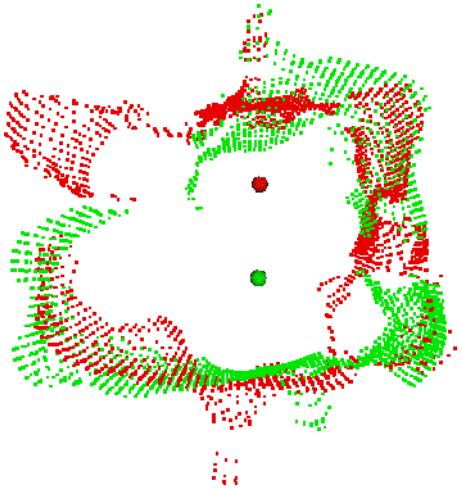
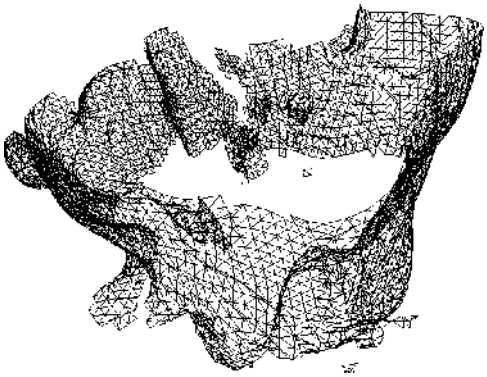


Figure 12. Two representative panoramas of the vision lab (top). The result of culling of two data sets of the vision lab based on visibility (bottom).

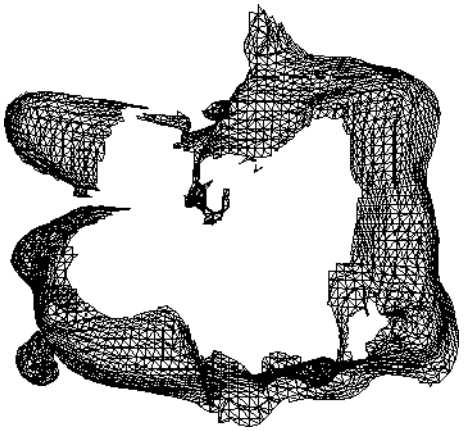
Points (top view)



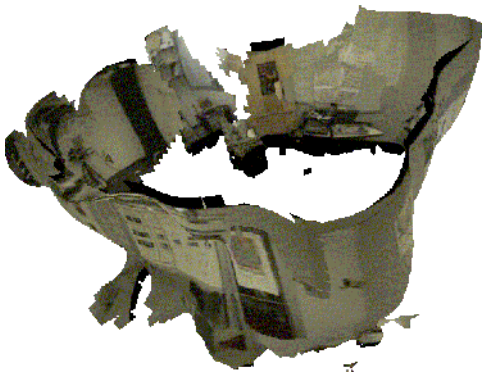
Surface (oblique view)



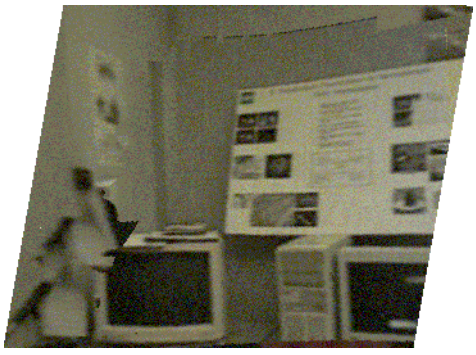
Surface (top view)



Textured Surface (oblique view)



Close Up #1 (max-blend)



Close Up #2 (max-blend)

Figure 13. the result of merging two textured 3-D data sets created with omnidirectional multi-baseline stereo of a lab. The texture of the merged model is created using the max texture blending scheme.

structures; each voxel structure contains the surface normal and probability information. By implementing the allocated voxels as a dictionary, the access to the voxel structure is made efficient. 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 in VRML. Some of the results from this paper can be viewed on web at *http://www.ius.cs.cmu.edu/usr/users/aej/www/research/vrml-modeling.html* using a VRML browser.

7. Conclusion

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 consider not only 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 only 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.

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 the 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, although 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 Smart Kiosk project at Cambridge Research Lab., Digital Equipment Corp. [28]. The Smart Kiosk can be described 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 this enhanced feature (using the recovered model of the environment) to the Smart Kiosk is currently underway.

It is perhaps worthwhile to investigate an alternative, view interpolation-based means of generating synthetic views of the model. However, these methods are not appropriate whenever 3-D structural information of the scene is desired or when certain kinds of views (such as fly-throughs involving camera positions very different than those of the known sampled views) are desired.

References

[1] 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.

[2] P. Besl and N. McKay. A method of registration of 3-D shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, no. 2, pp. 239-256, February 1992.

[3] J. Boissonnat. Geometric structures for three-dimensional shape representation. *ACM Trans. Graphics*, 3:4:266-286, October 1984.

[4] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, vol. 10, no. 3, pp. 145-155, April 1992.

[5] Y. Chen and G. Medioni. Surface description of complex objects from range images. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '94)*, 153-158, 1994.

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

[7] A. Elfies. Sonar-based real world mapping and navigation. *IEEE Jour. Robotics and Automation,* vol. RA-3, no. 3, pp. 249-265, 1987.

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

[9] J. Foley, A. van Dam, s. Feiner and J Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, New York, 1990

[10] J. Friedman, J. Bentley and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. on Mathematical Software*, 3:3, 209-226, September 1977.

[11] 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.

[12] A Hilton, A. Stoddart, J. Illingworth and T Windeatt. Reliable surface reconstruction from multiple range images. *Fourth European Conf. on Computer Vision (ECCV `96)*, pp. 14-18, April 1996.

[13] H. Hoppe, T. DeRose, T. DuChamp, J. McDonald and W. Stuetzle. Surface reconstruction from unorganized points. *Computer Graphics (SIGGRAPH '92)*, pp. 71-78. July 1992.

[14] A. Johnson and S. Kang. Registration and integration of textured 3-D data. *Digital Equipment Corp. Cambridge Research Lab.*

Tech. Report CRL 96/4, October 1996.

[15] S.B. Kang and R. Szeliski, 3-D scene data recovery using omnidirectional multibaseline stereo. *Conf. on Computer Vision and Pattern Recognition (CVPR '96)*, pp. 364-370, June 1996.

[16] C. Kolb. *Rayshade User Guide and Reference Manual*. August 1994.

[17] W. Lorensen and H. Cline. Marching Cubes: a high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH '87),* pp. 163-169, 1987.

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

[19] L. Matthies and S. Shafer. Error modeling in stereo navigation. *IEEE Jour. Robotics and Automation, vol.* RA-3, no. 3, pp. 239-248, June 1987.

[20] C. Montani, R. Scateni and R. Scopigno. A modified look-up table for implicit disambiguation of Marching Cubes. *Visual Computer*, vol. 10, pp. 353-355, 1994.

[21] S. Naher and C. Uhrig. *The LEDA User Manual.* May 1996.

[22] M. Rutishauser, M. Stricker and M. Trobina. Merging range images of arbitrarily shaped objects. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '94)*, 573-580, 1994.

[23] H.-Y. Shum, K. Ikeuchi and R. Reddy. Principal component analysis with missing data and its application to object modeling. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR-94)*, pp. 560-565, June 1994.

[24] M. Soucy and D. Laurendeau. Multi-resolution surface modeling from multiple range views. *Proc. IEEE Computer Vision and Pattern Recognition (CVPR '92)*, 348-353, 1992.

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

[26] G. Turk and M. Levoy. Zippered polygonal meshes from range images. *Computer Graphics (SIGGRAPH '94)*, pp. 311-318, 1994.

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

[28] K. Waters, J. Rehg, M. Loughlin, S.B. Kang, and D. Terzopoulos, Visual sensing of humans for active public interfaces. *Workshop on Computer Vision in Man-machine Interfaces*, Cambridge, UK, Apr. 1996.

[29] M. Wheeler. Automatic modeling and localization for object recognition. *Carnegie Mellon Univ., School of Computer Science Tech. Report CMU-CS-96-118*, October 1996.

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