

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 that the use of color decreases registration error by an order of magnitude. 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. .

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 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 volumetric multi-view merging algorithm that integrates the shape *and* appearance of complex scenes.

There has been much research in the area of model creation through multiple view merging. Shum et. al. [15], for example, recover the merged model through simultaneous determination of planar surface parameters and pose of constituent range data sets. They assume, however, that the surfaces of objects can be represented using planar patches. There have been many volumetric approaches to 3-D data integration [2][5][6][17], but none of them were designed to merge texture as well as shape. Furthermore, these algorithms assume that the views have already been aligned. Our approach builds on these approaches to solve the problem of merging shape and texture for generation of more descriptive scene models; our algorithm also includes a method for accurately registering 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 [8]. 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

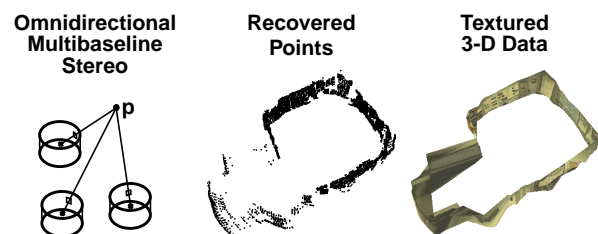


Figure 1. Shape recovery using omnidirectional multibaseline stereo.

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 1. We provide only a brief outline of the approach here; full details can be found in [8]. 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

The technique that we use to register all the 3-D data sets is essentially a modification of the Iterative Closest Point algorithm (ICP) [1]. In addition to using k-d trees for efficient closest point computations and a dynamic distance threshold [18], our algorithm uses shape and color information to improve the registration beyond that obtained with an ICP algorithm that uses just shape information. We call this variant the *Color ICP* technique; its concept is shown in Figure 2.

During integration of textured 3-D data, shape as well as texture are combined to form what is called 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 our algorithm to be able to blend texture correctly,

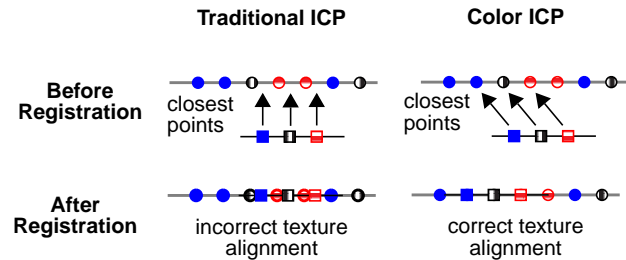


Figure 2. Demonstration of the use of color in registration. In traditional ICP closest points depend only on shape, so incorrect texture registration can occur. Since closest points depend on color and shape in Color ICP, it will align texture correctly.

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 error 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 [8]) 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. This is accomplished by modifying the distance metric used in the closest point computation between points $\mathbf{p}_1 = (x_{11}, x_{12}, x_{13}, c_{11}, c_{12}, c_{13})$ and $\mathbf{p}_2 = (x_{21}, x_{22}, x_{23}, c_{21}, c_{22}, c_{23})$ to include both shape and color information (x 's and c 's respectively), i.e.,

$$d_6(\mathbf{p}_1, \mathbf{p}_2) = \left[\begin{array}{l} (x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + \\ (x_{13} - x_{23})^2 + \Gamma_1(c_{11} - c_{21})^2 + \\ \Gamma_2(c_{12} - c_{22})^2 + \Gamma_3(c_{13} - c_{23})^2 \end{array} \right]^{\frac{1}{2}} \quad (1)$$

where $\Gamma = (\Gamma_1, \Gamma_2, \Gamma_3)$ are scale factors that weigh the importance of color against the importance of shape. In Color ICP, a six dimensional k-D tree is used to speed up closest point computations.

A 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 a synthetic textured 3-D data set (where the correct registration is known) is shown for the traditional ICP and Color ICP algorithm in Figure 3. It illustrates the order of magnitude improvement in registration using color information in addition to 3-D shape, making integration of texture feasible.

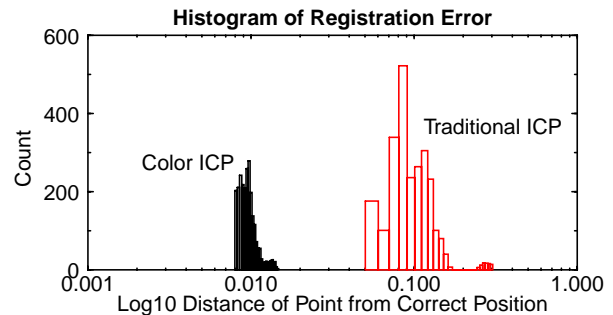


Figure 3. Histogram of registration errors for the traditional ICP and Color ICP algorithms. The histogram clearly shows that the Color ICP algorithm is an order of magnitude improvement over the traditional ICP algorithm.

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 [3][11] 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.

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 [12]. 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 .

$$G(\lambda, \sigma_\alpha, \sigma_\beta, \sigma_\gamma, \sigma_\delta) = \lambda G_E(\sigma_\alpha, \sigma_\beta) + (1 - \lambda) G_S(\sigma_\gamma, \sigma_\delta) \quad (2)$$

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.

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

$$G_E(\alpha, \beta, \sigma_\alpha, \sigma_\beta) = \frac{1}{\sqrt{2\pi}} \sqrt{\frac{1}{\sigma_\alpha^2} + \frac{1}{\sigma_\beta^2}} \exp\left(-\frac{1}{2}\left(\frac{\alpha^2}{\sigma_\alpha^2} + \frac{\beta^2}{\sigma_\beta^2}\right)\right) \quad (3)$$

$$\alpha = \sqrt{\mathbf{x} \cdot \mathbf{x} - \mathbf{x} \cdot \hat{\mathbf{v}}} \quad \beta = \mathbf{x} \cdot \hat{\mathbf{v}}$$

$$\mathbf{x} = \mathbf{Q} - \mathbf{P} \quad \hat{\mathbf{v}} = \frac{(\mathbf{S} - \mathbf{P})}{\|\mathbf{S} - \mathbf{P}\|}$$

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. \mathbf{S} , \mathbf{P} and \mathbf{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, 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 shown in Figure 4.

Matthies and Shafer [12] 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 \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} for M is calculated as follows:

$$\bar{d} = \frac{1}{N} \sum_{\mathbf{p} \in L_P} \|\mathbf{p} - \mathbf{S}\| \quad d_P = \frac{1}{N_P} \sum_{\mathbf{p} \in L_P} \bar{d} - \|\mathbf{p} - \mathbf{S}\| \quad (4)$$

$$d_{rms} = \frac{1}{N_M} \sqrt{\sum_{\mathbf{p} \in M} d_P^2}$$

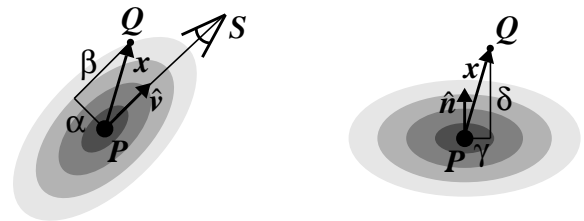
d_{rms} measures the average local change in distance along the viewing direction which is a good measure of sensor error assuming that neighborhoods are locally planar, with normals roughly oriented along the viewing direction. The variances in the sensor error model are set automatically based on estimated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

$$G_S(\gamma, \delta, \sigma_\gamma, \sigma_\delta) = \frac{1}{\sqrt{2\pi}} \sqrt{\frac{1}{\sigma_\gamma^2} + \frac{1}{\sigma_\delta^2}} \exp\left(-\frac{1}{2}\left(\frac{\gamma^2}{\sigma_\gamma^2} + \frac{\delta^2}{\sigma_\delta^2}\right)\right) \quad (5)$$

$$\gamma = \sqrt{\mathbf{x} \cdot \mathbf{x} - \mathbf{x} \cdot \hat{\mathbf{n}}} \quad \delta = \mathbf{x} \cdot \hat{\mathbf{n}} \quad \mathbf{x} = \mathbf{Q} - \mathbf{P}$$

where \mathbf{g} is the distance of the query point \mathbf{x} from the unit



Sensor Error Model G_E

Point Spread Model G_S

Figure 4. 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.

surface normal \hat{n} and d 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_g^2 the variance along the tangent plane and σ_d^2 the variance along the surface normal. A 2-D slice of the surface spreading geometry is given in Figure 4.

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

$$r_{av} = \frac{1}{N_E} \left(\sum_{p_i, p_j \in M} \|p_i - p_j\| \right). \quad (6)$$

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

An example of the recovered surface probability distribution is shown in Figure 7(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 5, 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 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 (g) and the consensus surface normal (n) are computed at each voxel. As shown in Figure 5, 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. 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 [10] with a modified lookup table of 22 cases to prevent the creation of holes in the surface [13]. The consensus surface mesh generated from the 6 synthetic data sets is shown in Figure 7. 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

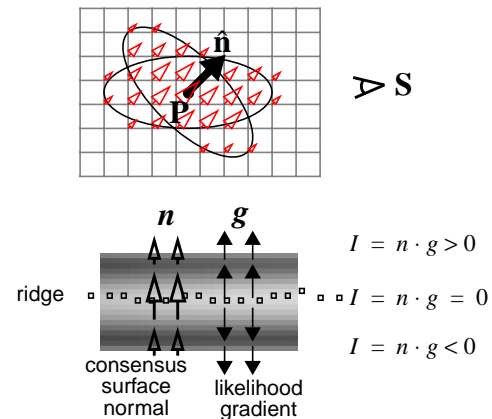


Figure 5. The occupancy grid stores consensus surface normal and is updated by each data point using vector addition (top). The consensus surface normal defines the direction along which to search for local maxima in surface likelihood. (bottom).

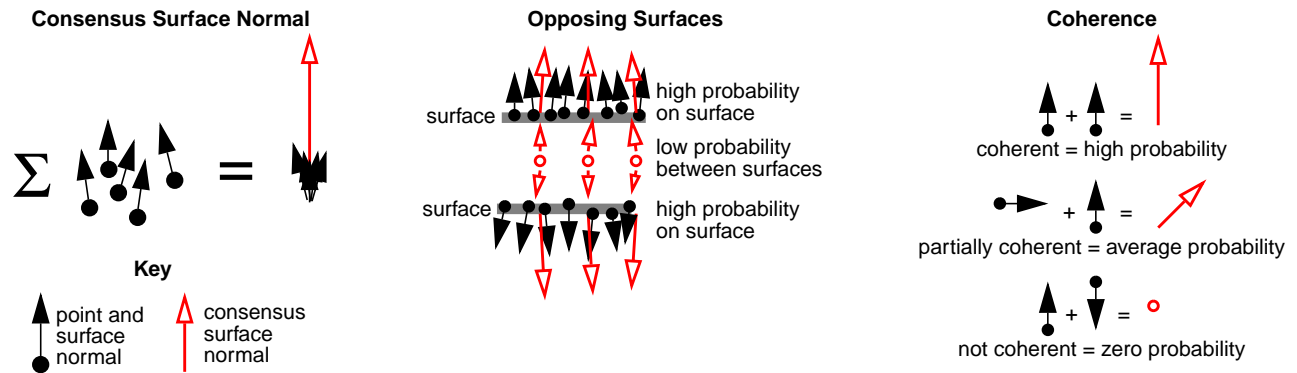


Figure 6. Consensus surface normal definitions. The consensus surface normal is the weighted sum of the normals of surrounding points (left). Adding probabilities as vectors prevents opposing surfaces from mixing (middle). Coherence of normals determines magnitude of consensus surface normal (right).

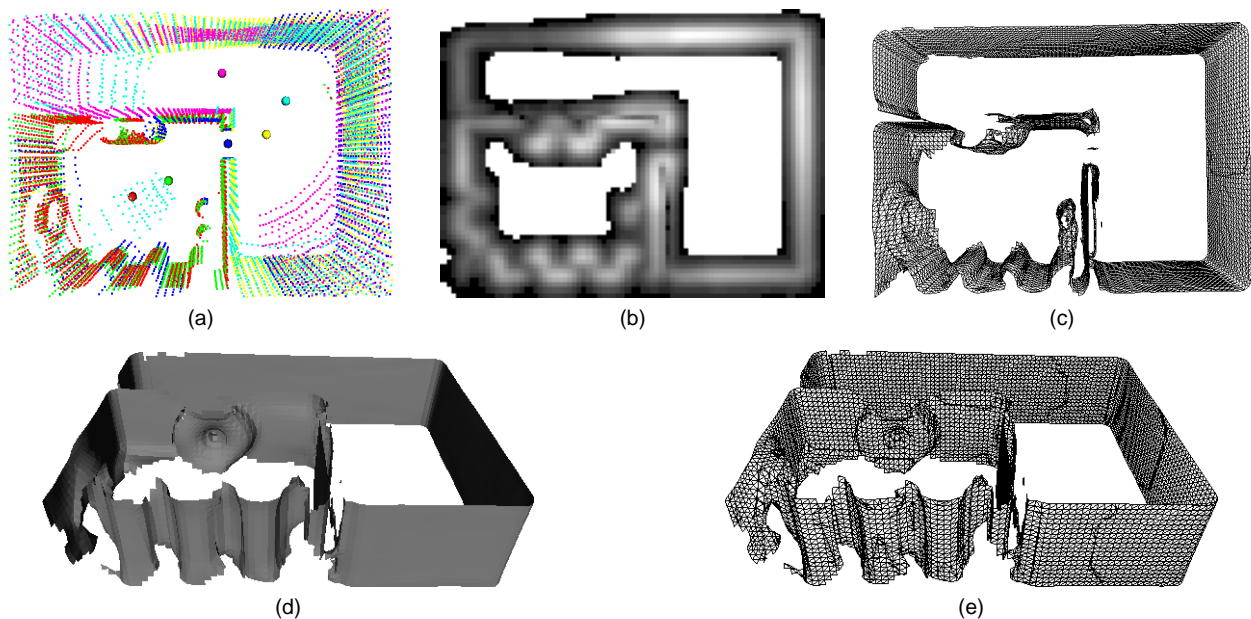


Figure 7. Registered points sets (a) with sensor origins shown as shaded spheres and the middle horizontal slice of surface probability through the voxel space for those points (b). Notice that only allocated voxels are shown. Three views of the consensus surface mesh generated for six registered data sets are also shown (c)-(e).



Figure 8. Two representative panoramas of the vision lab.

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 \mathbf{n}_i in each voxel, and those vectors 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

$$w_i = \max(0, \mathbf{n}_i \cdot \mathbf{n}_c). \tag{7}$$

If w_i 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, 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 Marching Cubes, 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.

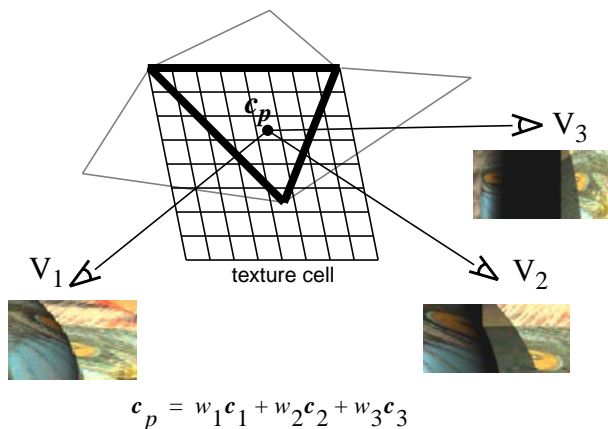


Figure 9. The geometry for creating a texture map cell for a face. The color of each pixel of the cell is the weighted average of the colors projected onto it by data sets that view the pixel. Each face in the consensus surface mesh has an associated texture cell.

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. Alternatively, the texture at a pixel in a cell can be set to the texture projected from the data set with maximum texture weight. The geometry of texture blending is illustrated in Figure 9. Figure 10 shows the result of texture blending six textured 3-D data sets.

We have also worked with real scenes; results of one experiment with a real office scene are shown in Figure 11.

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 8.

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.

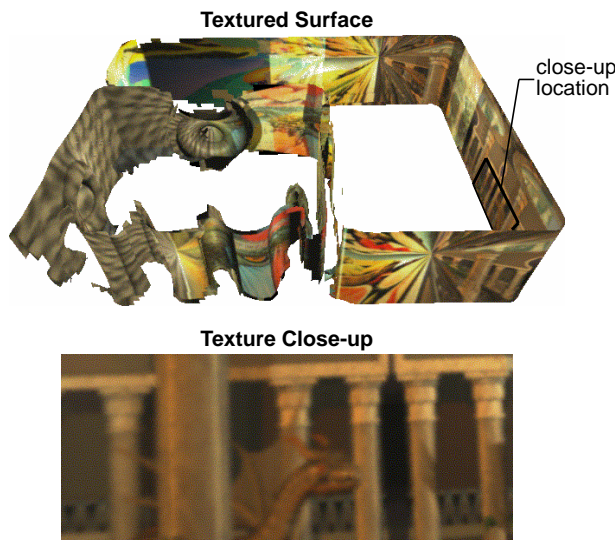


Figure 10. The result of integrating six textured 3-D data sets created directly from a synthetic room model. The complete room model with texture blended on the surfaces of the room is shown as well as a close up of the texture blending.

5. Implementation

The code for our model merging work is written in C++ and uses the Library of Efficient Data types and Algorithms (LEDA) [14]. 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. 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.

6. 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 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

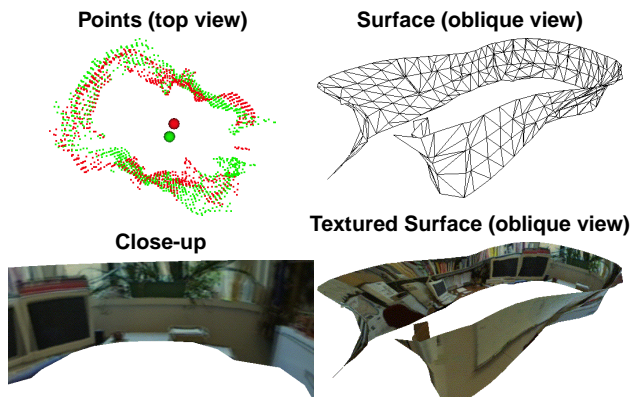


Figure 11. The result of integrating two textured 3-D data sets created with omnidirectional stereo of an office. The registered points, wire frame surface, texture mapped surface and close-up of the texture mapping are shown. Note: the two small spheres in the top view of the 3-D points represent the different camera center locations.

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. [16]. 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.

7. Summary

We have described our approach to merging multiple

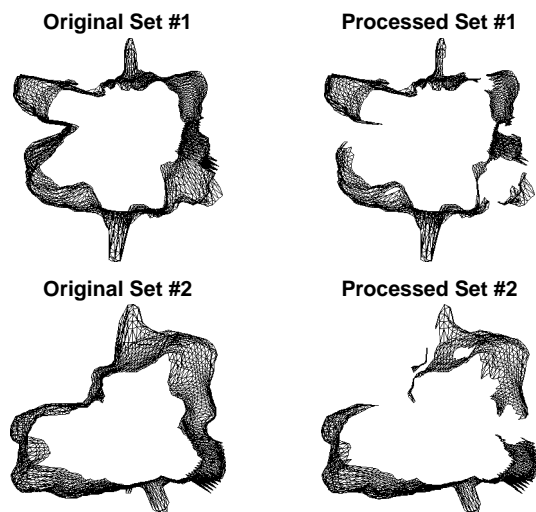


Figure 12. The result of culling of two data sets of the vision lab based on visibility. Left: original surface meshes, right: culled surface meshes.

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.

A more detailed description of our work on merging different textured 3-D data sets can be found in [7].

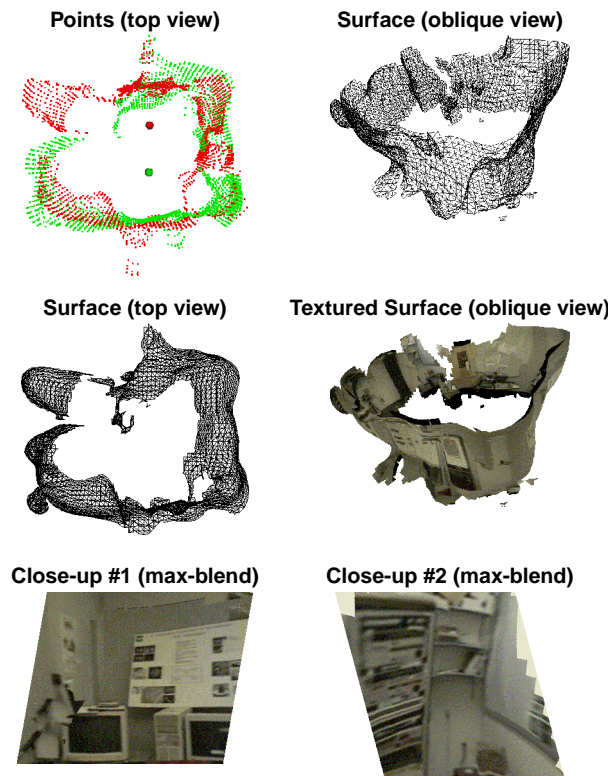


Figure 13. the result of merging two textured 3-D data sets created with omnidirectional multi-baseline stereo of a lab. The texture of the merged model is created using the max texture blending scheme.

References

- [1] 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.
- [2] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Computer Graphics (SIGGRAPH '96)*, August 1996.
- [3] A. Elfies. Sonar-based real world mapping and navigation. *IEEE Jour. Robotics and Automation*, vol. RA-3, no. 3, pp. 249-265, 1987.
- [4] 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.
- [5] A. Hilton, A. Stoddart, J. Illingworth and T. Windett. Reliable surface reconstruction from multiple range images. *Fourth European Conf. on Computer Vision (ECCV '96)*, pp. 14-18, April 1996.
- [6] 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.
- [7] 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.
- [8] S. Kang, and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. *Digital Equipment Corporation Cambridge Research Lab. Tech. Report CRL 95/6*, October 1995.
- [9] C. Kolb. *Rayshade User Guide and Reference Manual*. August 1994.
- [10] W. Lorensen and H. Cline. Marching Cubes: a high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH '87)*, pp. 163-169, 1987.
- [11] M. Martin and H. Moravec. Robot Evidence Grids. *Carnegie Mellon University Robotics Institute Tech. Report CMU-RI-TR-96-06*, March 1996.
- [12] 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.
- [13] 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.
- [14] S. Naher and C. Uhrig. *The LEDA User Manual*. May 1996.
- [15] 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.
- [16] K. Waters, J. Rehg, M. Loughlin, S. Kang and D. Terzopoulos. *Visual sensing for humans for active public interfaces*. Cambridge, UK, 1996
- [17] M. Wheeler. Automatic modeling and localization for object recognition. *Carnegie Mellon Univ., School of Computer Science Tech. Report CMU-CS-96-118*, October 1996.
- [18] 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.