LaTeX2e cmu-techreport

Image-Based Multiresolution Modeling by Surface Deformation

Li Zhang, Steven M. Seitz

CMU-RI-TR-00-07

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

March, 2000

(C) 2000, Li Zhang, Steven M. Seitz. All rights reserved.

Abstract

This paper presents a novel approach for constructing multiresolution surface models from a set of calibrated images. The output is a texture-mapped triangular surface mesh that best matches all the input images. The mesh is obtained by deforming a generic initial mesh such as a sphere or cube according to image and geometry-based forces. This technique has the following key features: (1) the initial mesh is able to converge to the object surface from arbitrarily far away, (2) the resolution of the final mesh adapts to the local complexity of the object, (3) sharp corners and edges of object surface are preserved in the final mesh, (4) occlusion is correctly modeled during convergence, (5) re-projection error of the final mesh is optimized, (6) the output is ideally suited for rendering by existing graphics hardware. The approach is shown to yield good results on real image sequences.

1 Introduction

A fundamental problem in computer graphics is obtaining high quality textured surface models. The ability to construct such models from photographs is extremely attractive, due to the ease with which high-resolution digital images may be obtained and processed. While the last few years have seen great progress in image-based 3D reconstruction techniques in the computer vision community [4, 3, 2, 6], most current methods focus on creating point clouds rather than surface models directly. In contrast, most graphics architectures require surface representations. Consequently, point cloud data is typically converted to a surface model such as a triangular mesh or NURBS surface before rendering. Due to the lack of connectivity information, this conversion process is difficult and error-prone [7, 8]. Rather than creating a point-based representation as an intermediate step, this paper advocates reconstructing a texture-mapped triangular surface mesh directly from a sequence of images.

While accurate modeling of geometry and reflectance can help to produce realistic computer graphics, recent work in image based rendering [9, 10, 11, 12] has demonstrated that accurate geometry is not essential. Rather, the following characteristics are desirable:

- Photorealism: renderings of the model should appear indistinguishable from photographs of the true scene;
- Compactness: the model should have no more detail than is necessary to adequately reproduce the photographs.

In this paper we introduce a surface-based reconstruction technique that is specifically designed to optimize these two objectives. The approach works by deforming a generic initial mesh such as a sphere or cube according to image- and geometry-based forces. Attaining these goals requires solving several difficult problems. First, unlike most approaches, we do not assume a priori knowledge of depth or image correspondence information. Rather, correspondence information is robustly and automatically recovered as a by-product of the mesh optimization procedure. Second, the initial surface may be arbitrarily far away from the true scene and the approach must model occlusions correctly to enable convergence. Third, sharp corners and edges of the scene should be preserved in the reconstruction. And finally, renderings of the resulting mesh should faithfully reproduce the input photographs. The method that we present is shown to have all of these properties.

Our technique may be thought of as a generalization of image mosaics[13]. In image mosaicing, a set of input images are projected onto a cylinder, sphere, or plane. The warping transformation is estimated by minimizing the intensity error between projected images. In our surface reconstruction algorithm, the input images are projected onto a triangular mesh. The mesh is iteratively deformed to the object surface by minimizing the intensity error between projected images. However, our problem is much more difficult than image mosaicing. Because the surface geometry

is unknown a priori, new vertices have to be inserted into the mesh and redundant vertices have to be removed from the mesh during deformation in such a way that the mesh is adaptive to the object surface.

The reminder of the paper is organized as follows. Section 2 gives an overview of our algorithm. Section 3 describes the steps of our surface deformation algorithm. Section 4 presents the results on both synthesized and real images. We close with a summary and a description of future work.

1.1 Previous Work

Several promising techniques have recently been proposed for reconstructing accurate 3D models from multiple images. The work most related to ours is as follows.

Fua and LeClerc [2] pioneered the mesh-based framework for scene reconstruction. Their approach optimizes an initial mesh according to an objective function taking into account multi-image correlation, surface shading and geometric smoothness. Their optimization is based on the conjugate gradient algorithm and uses image derivatives. In order for the algorithm to converge, the vertices of initial mesh must be projected to within a few pixels of their true locations. A dense depth map from stereo algorithm [1] is used to initialize such a mesh. Moreover, the number of vertices doesn't change during their optimization procedure, which is not ideal for capturing complex scene with multiple resolution.

Faugeras and Keriven [3] propose a Level Set approach for surface reconstruction. They represent a surface implicitly as the zero set of a function over $\mathbf{R}^3$. Their approach maximizes a functional based on pairwise image correlation using variational methods. The object surface is assumed to be represented by the function that maximizes the functional. Although they obtain good results and can handle topological changes, their procedure is computationally expensive and has no known convergence properties. Moreover, their algorithm operates at a fixed resolution and doesn't exploit graphics hardware.

Seitz and Dyer [4] and Kutulakos and Seitz [5] introduce voxel-based based approaches for recovering shapes of arbitrary geometry and topology. Because all the voxels must be visited in a monotonic order, their technique cannot recover if a voxel is carved away incorrectly. Furthermore, the resolution of the output shape must be specified a priori and the recovered shape will have a voxelated appearance. While these techniques produce a shape with error below a specified threshold, they do not attempt to minimize re-projection error. Voxel models are not as efficeint for modeling flat surfaces as mesh models.

Hoppe et al. [7, 8] describe an algorithm that takes as input an unorganized set of points and produces as output a mesh that best approximate these points. Their approach is similar to ours, but it is not image-based and instead requires accurate range data as input.

In this paper, we present a multiresolution surface reconstruction algorithm that converges robustly and accurately even when the surfaces of the initial mesh are far

from the true scene. Unlike previous approaches, our method optimizes reprojection error and automatically adapts to match the local complexity of the scene.

2 Overview

Our overall objective is to generate a compact 3D surface that is optimally consistent with a set of input images, and to achieve this goal by deforming an initial surface mesh. We have chosen the triangular mesh as a basic surface representation because: (1) it is a first order approximation of any arbitrary surface, (2) its multiresolution properties have been extensively studied [15], and (3) polygonal meshes are well suited for processing with the standard graphics hardware that comes pre-packaged with todays desktop PC's. A triangular mesh M consists of triangles T, edges E and vertices V. Each vertex $\mathbf{v} \in V$ is a point in 3D space. Each edge $e \in E$ has two vertices in V. Each triangle $t \in T$ in has three edges in E.

2.1 Mesh Consistency

In order to assess the quality of a reconstruction, we have to measure to what extent a mesh is *consistent* with a set of input images. In this paper, we assume the scene is approximately Lambertian and use the variance of colors from multiple images as a consistency metric. To be specific, suppose that $\mathbf{p}$ is a point on the mesh that is visible in images $I_1, I_2, \ldots, I_N$, and $\mathbf{p}$ is assigned colors $c_1, c_2, \ldots, c_N$. Following [4], we use

$$\sigma_p{}^2 = \sum_{i=1}^{N} (c_i - \bar{c})^2$$

as the consistency metric for $\mathbf{p}$. A consistency metric for the mesh is obtained by integrating $\sigma_p{}^2$ over all points on the surface. To account for image discretization, we instead sample the surface of the mesh at points $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_S$ and use

$$\Phi = \sum_{i=1}^{S} \sigma_{p_i}{}^2$$

as the consistency metric for the mesh. The details in sampling the mesh are presented in Section 3.2.

2.2 Mesh Deformation

We seek to optimize the mesh to minimize Φ. Our approach is to compute, for each sampling point $\mathbf{p}_i$, the displacement that minimize $\sigma_{p_i}{}^2$. The result is a *surface flow* which describes how points on the current surface should move. By analyzing the surface flow we can deduce how the current mesh should deform in order to decrease Φ. The flow is computed using a gradient-based procedure, described in Section 3.3. Mesh deformation is accomplished by adding and moving vertices in a scene-dependent manner in accordance with the computed surface flow. More specifically, we wish to minimize the sum of distances between the displaced sampling

points and the deformed mesh. The procedures for moving and adding vertices are described in Section 3.4 and Section 3.5 respectively.

2.3 Mesh Simplification

After deformation, some parts of the mesh surface may become flat while others may have a dense cluster of vertices. In both cases, the mesh can be simplified with minimal effect on re-projection quality. We therefore wish to represent the flat surface with fewer vertices and to collapse superfluous edges. Both fo these two goals are achieved by our mesh simplification procedure, described in Section 3.6.

Overall, the steps of our mesh deformation algorithm are outlined as follows:

initialize a mesh to contain the scene;
do {

1. **sample points on the mesh;**
2. **calculate the displacements of samples;**
3. **deform and subdivide the mesh to fit the displaced samples;**
4. **simplify the mesh;**

} **until the consistency metric Φ is optimized.**

Figure 1 gives an intuitive illustration of our algorithm.

3 Surface Deformation

The following subsections correspond to the steps of our surface deformation algorithm outlined above.

3.1 Mesh Initialization

In Space Carving [5], the initial scene volume is proved to converge to the true scene if the true scene lies completely inside the initial scene volume. The mesh surface can be thought of as the boundary of its interior volume. Therefore, the true scene is required to lie inside the initial mesh surface.

3.2 View-Dependent Sampling

Before deforming the mesh, we first project all the images onto the mesh. In this paper, we set up a sampling grid for each triangle $t \in T$ to sample the projected images on the mesh, as shown in Figure 2(a). When an input image is projected onto triangles, the oblique or far-away triangles are colored with less effective resolution, shown in Figure 2(b). Because different triangles in the mesh may have different

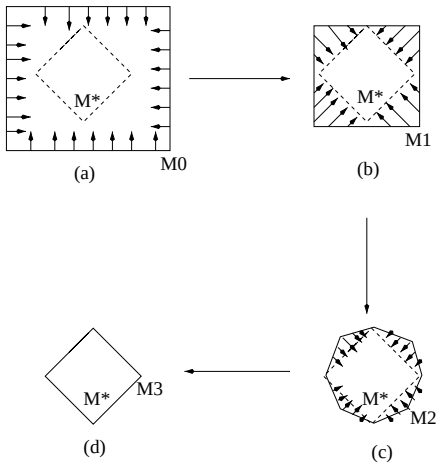


Figure 1: Illustration of Surface Deformation: M* is the object surface; M0, M1, M2, and M3 are the deforming mesh; small arrows are the displacements of sample points. (a) M0 is the initial mesh with M* inside; (b) M1 is obtained by deforming M0; (c) M2 is obtained by deforming and subdividing M1; (d) M3 is obtained by deforming and simplifying M2, and M3 coincides with M*.

locations and orientations relative to the image plane of input views, it is desirable that they have different sampling resolutions according to the input viewpoint configuration and image resolution. For a particular triangle, it may be colored by multiple images from different viewpoints with different effective resolution. We want to sample the triangle with the highest effective coloring resolution. To be more specific, we set, for each triangle $t \in T$, the sampling resolution r_t such that the number of sampling points inside t equals the maximum number of pixels inside the projected t on different image planes. In practice, the number of pixels inside the projected t is approximated as half of the number of pixels inside the bounding rectangle of projected t.

3.3 Surface Flow

When a facet of the current mesh is close to the object surface, the desired displacement of each sample point may be estimated by moving toward the negative gradient of ${\sigma_p}^2$, i.e., in the direction of $-\frac{\partial {\sigma_p}^2}{\partial p}$. However, ${\sigma_p}^2$ is a nonlinear function of $\mathbf{p}$ and moving $\mathbf{p}$ based on the gradient is subject to local minimum. For practice, the gradient information is unreliable when $\mathbf{p}$ is far away from the object surface.

In order to handle the case when the mesh is still far away from the object surface we want to model, we initialize a mesh such that the object is inside the mesh. For a particular sampling point $\mathbf{p}$ with outward normal $\mathbf{n}_p$ on the mesh, we assume that $\mathbf{p}$ is far from object surface if σ_p is large and $\mathbf{p}$ is close to object surface

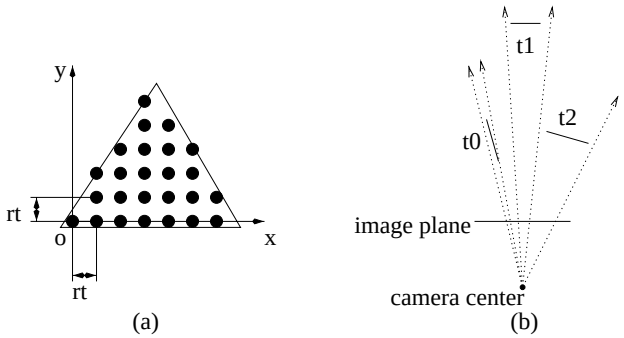


Figure 2: (a) Sampling grid for triangle t with sampling resolution rt; (b) different relative configuration between triangles and camera: t0 is oblique, t1 is far-away, t2 is typical. t0, t1 and t2 should have different sampling resolution.

if σ_p is small. According to Space Carving [5], we should move $\mathbf{p}$ along $\mathbf{n}_p$ to carve it away if it is still far away from the object surface. As a result, our strategy in this paper is to move $\mathbf{p}$ more along $-\frac{\partial {\sigma_p}^2}{\partial p}$ if σ_p is small and to move $\mathbf{p}$ more along $-\mathbf{n}_p$ if σ_p is large. Both of these conditions are met by expressing the displacement $\delta\mathbf{p}$ of sampling point $\mathbf{p}$ by the equation:

$$\delta\mathbf{p} = -(1 - {\sigma_p}^2)\frac{\partial {\sigma_p}^2}{\partial p} - {\sigma_p}^2\mathbf{n}_p$$

where ${\sigma_p}^2$ is the normalized variance of colors.

By setting $\delta\mathbf{p}$ in this manner, the mesh acts as a Space Caving techique when far away from the object surface and as a gradient-based technique when close to the object surface. Therefore, the mesh is converges robustly and accurately with little initialization requirement.

3.4 View-Dependent Mesh Deformation

The surface flow defines a displacement field over the surface of the mesh. We now describe how to update the mesh to best approximate the flow.

Each sampling point $\mathbf{p}$, with its displacement $\delta\mathbf{p}$, belongs to some triangle $t \in T$ in the mesh. Suppose that t consists of vertices $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$. Let $\mathbf{p}^*$ be the orthogonal projection of $\mathbf{p} + \delta\mathbf{p}$ on t. We may express $\mathbf{p}^*$ in barycentric coordinates as:

$$\mathbf{p}^* = b_0\mathbf{v}_0 + b_1\mathbf{v}_1 + b_2\mathbf{v}_2$$

We want to minimize the sum of squared distances of between the displaced samples and the deformed mesh, expressed as:

$$\Delta = \sum\nolimits_{p \in P} \|\mathbf{p} + \delta\mathbf{p} - \mathbf{p}^*\|^2$$

where P is the set of all sampling points on the mesh surface.

If we move $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ by $\delta\mathbf{v}_0, \delta\mathbf{v}_1, \delta\mathbf{v}_2$, we introduce a displacement of $\mathbf{p}^*$, which can be approximated by

$$\delta\hat{\mathbf{p}}^* = b_0\delta\mathbf{v}_0 + b_1\delta\mathbf{v}_1 + b_2\delta\mathbf{v}_2$$

This is only an approximation of the actual $\delta\mathbf{p}^*$ because $\{b_0, b_1, b_2\}$ changes as $\{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2\}$ changes. However, this approximation is efficiently computed and performs well in practice. Therefore our problem is to find $\delta\mathbf{v}$ for each $\mathbf{v}$ such that Δ is minimized. The conjugate gradient method [19] can be used for this purpose. However, it requires matrix manipulation and has complexity that is cubic in the number of vertices. We instead use the following iteration equations to move $\mathbf{v}$:

$$\mathbf{v}^{(0)} = \mathbf{v}$$

$$\delta\mathbf{v}^{(k)} = \frac{\Sigma_{\mathbf{v}\in t}\Sigma_{\mathbf{p}\in t}(\mathbf{p} + \delta\mathbf{p} - \mathbf{p}^{*(k)})b_v(\mathbf{p}^{*(k)})}{\Sigma_{\mathbf{v}\in t}\Sigma_{\mathbf{p}\in t}b_v(\mathbf{p}^{*(k)})}$$

$$\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + \delta\mathbf{v}^{(k)}$$

where $\mathbf{p}^{*(k)}$ is the projection of $\mathbf{p}+\delta\mathbf{p}$ on $t^{(k)} = \{\mathbf{v}_0{}^{(k)}, \mathbf{v}_1{}^{(k)}, \mathbf{v}_2{}^{(k)}\}$ and $b_v(\mathbf{p}^{*(k)})$ is the barycentric coordinate of $\mathbf{p}^{*(k)}$ with respect to $\mathbf{v}^{(k)}\in t^{(k)}$. Our iteration equations converge after less than ten steps in practice.

View-dependent sampling may be taken into account by defining

$$\Delta_t = \frac{1}{{r_t}^2}\Sigma_{\mathbf{p}\in t}\|\mathbf{p} + \delta\mathbf{p} - \mathbf{p}^*\|^2$$

where r_t is the sampling resolution of triangle t introduced in Section 3.2. We want to find a $\delta\mathbf{v}$ for each $\mathbf{v}\in V$ to minimize:

$$\Delta = \Sigma_{t\in T}\Delta_t$$

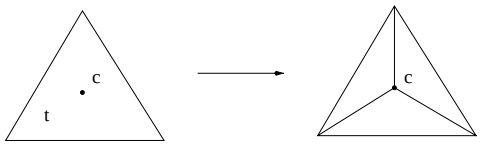
Similar iteration equations can be easily derived for this view-dependent case.

3.5 Mesh Subdivision

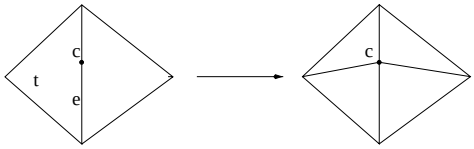
Each triangle t may contain a large number of sampling points, but has only three vertices, i.e., 9 degrees of freedom (DOF). The displacement field on t may not be well-approximated by a plane. Figure 1(b) gives an example in the 2D case, where the displacements of two endpoints are not sufficient to fit the edge with the displacements of the samples. It would be better to split the triangle or edge by inserting a new vertex and then to refine the approximation.

Specifically, we first select the triangle $t\in T$ with the largest Δ_t. Then we calculate the centroid, $\mathbf{c}_t$, of the projected displaced samples weighted by their residual errors:

$$\mathbf{c}_t = \frac{\frac{1}{{r_t}^2}\Sigma_{p\in t}\|\mathbf{p} + \delta\mathbf{p} - \hat{\mathbf{p}^*}\|^2\mathbf{p}^*}{\Delta_t}$$



(a) split triangle t at c



(b) split edge e of triangle t at c

Figure 3: A new vertex c is inserted in the mesh. (a) if c is inside a triangle t, t is splitted; (b) if c is close to an edge e, e is splitted.

Finally, we split the triangle t by inserting a new vertex at $\mathbf{c}_t$, as shown in Figure 3(a). As a special case, we split one edge $e \in t$ at $\mathbf{c}_t$ if $\mathbf{c}_t$ lies close to that edge, as shown in Figure 3(b). To make the optimization more robust, we perform the following local re-triangulation after each triangle or edge split.

Suppose that $\mathbf{c}$ is a newly inserted vertex with vertices $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\}$ and $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ around it, as illustrated in Figure 4. For each edge $\{\mathbf{u}_i, \mathbf{u}_{i+1}\}$, we check if $\angle \mathbf{u}_i\mathbf{c}\mathbf{u}_{i+1} + \angle \mathbf{u}_i\mathbf{v}_i\mathbf{u}_{i+1} > 180°$. If so, we split $\{\mathbf{u}_i, \mathbf{u}_{i+1}\}$ at the intersection $\mathbf{w}_i$ of $\mathbf{u}_i\mathbf{u}_{i+1}$ and $\mathbf{c}\mathbf{v}_i$.

Combined with mesh deformation, our mesh subdivision algorithm works as follows:

do {

1. **deform the subdivided mesh by minimizing Δ;**

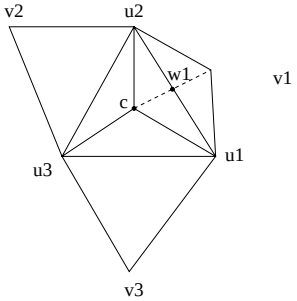


Figure 4: In local retriangulation around c, edge u1,u2 is splitted at w1. Dotted line are new edges due to edge split

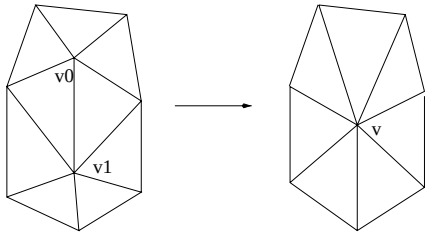


Figure 5: edge collapse, $(v0, v1) \rightarrow v$

2. **select the triangle t with the largest residual errors Δ_t;**

3. **split the t or one of its edges;**

4. **re-triangulate the mesh around the newly inserted vertex;**

} **until Δ can not be reduced significantly by inserting new vertex.**

3.6 Mesh Simplification

If we keep inserting new vertices into the mesh and deforming it, the mesh will become denser and denser. This is not desirable. First, small edges will introduce singular triangles into the mesh, which are not stable for numercical computation. Second, it is not necessary to represent a flat surface with a large number of vertices. These two problem could be avoided by simplifying the mesh within a certain error metric.

In our work, we use Garland and Heckbert's Quadric Error Metric [14] to simplify the mesh. The basic idea is to iteratively collapse an edge to a vertex if its length is too small or both its two vertices have low curvature. Figure 5 is an illustration of the edge collapse operation. Edges are ranked according to the amount of error introduced into the mesh as a result of edge collapsing. This error is expressed as the sum of squared distances of the new vertex to all triangle planes incident to the two end vertices of the collapsed edge. At each iteration, the edge with the smallest error is collapsed to a vertex.

We use Garland and Heckbert's algorithm because it is both fast and reasonably accurate. It removes undesired small edges and reduces the number of vertcies in flat patches of the mesh. In practice, it is very important to make our mesh stable and simplified with minimal geometry error introduced. Because our mesh has view-dependent resolution, we measure the distance between a vertex v and a triangle t in unit of r_t. As a result, our quadric error metric is also view-dependent.

4 Results

We now present some results obtained from both synthetic and real images.

4.1 Synthetic Images

We first model a smoothly textured cube by deforming an initial sphere of 256 vertices, with 26 calibrated images taken around the cube, four of which are shown in Figure 6(a). The wireframes of the sphere in different stages of deformation are shown in Figure 6(c). The final mesh contains 130 vertices. We render the texture-mapped output in different novel views, as shown in Figure 6(b).

For a simple shape, such as a cube, our algorithm can pretty accurately reconstruct its geometry, with only minor errors on the edge or corner. During convergence, our algorithm is able to reduce the number of vertices and simplify the mesh structure.

All the input images have the same resolution 128x128. Our sampling resolution is half the one we recommend in Section 3.2. That is, one sampling point on the mesh approximately corresponds to 4 image pixels. It takes our algorithm about five minutes to finish the whole recontruction procedure on a 450MHz HP Kayak PC Workstation.

4.2 Real Images

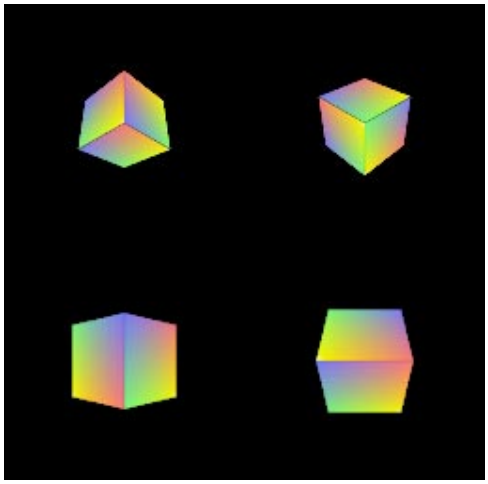
We then model a toy dinosaur by deforming an initial ellipsoid of 256 vertices, with 11 images taken above the head of the dinosaur, four of which are shown in Figure 6(d). The wireframes of the dinosaur in different stages are also shown in Figure 6(f). The final mesh contains 884 vertices. We also render the texture-mapped output in different novel views, as shown in Figure 6(e).

The dinosaur is a very complex shape with many concavities. Our algorithm can still model the overall shape well and give reasonable appearances from novel viewpoints. However, there are some artifacts in the reconstructed model due to interpenetration of faces. During convergence, our algorthm automatically increases the number of vertices to capture the variations in the shape of the object.

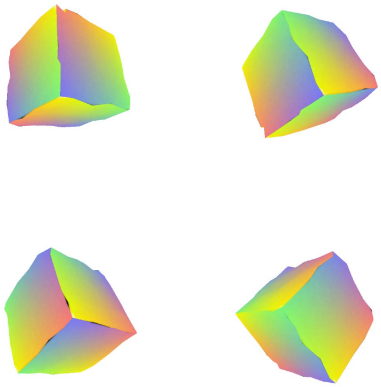
This set of dinosaur images is a subset of the 21 images used in Voxel Coloring [4]. All the input images have the same resolution 486x640. Our sampling resolution in this case is the same as recommended in Section 3.2. It takes our algorithm about 45 minutes to finish the whole reconstruction on the same platform above.

5 Summary and Future Work

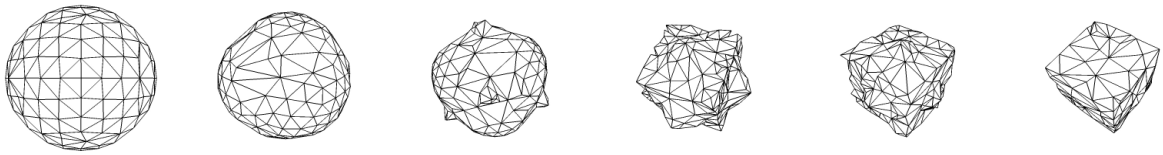
We have presented a novel approach for constructing multiresolution surface models from a set of calibrated images. It is based on surface deformation technique and converges with few requirements on the initial mesh. Visibility is taken into account during convergence. The final mesh models corners and edges accurately and automatically adapts to match the local complexity of the scene. Our technique produces a texture-mapped triangular surface, which is directly supported by existing graphics hardware.



(a)

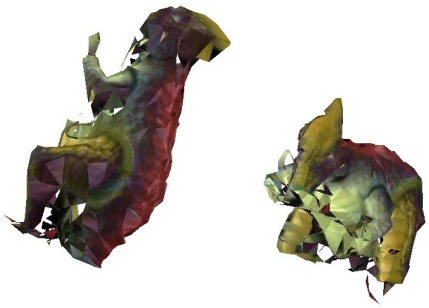


(b)



(c)

(d)



(e)

(f)

Figure 6: (a) four of the 26 synthetic images used to model a texture-mapped cube, (b) 4 images of the final texture-mapped cube from novel viewpoints, (c) the wireframes in different stages of deformation from a sphere to a cube, (d) four of the 11 real images used to model a toy dinosaur, (e) 2 images of the final texture-mapped dinosaur from novel viewpoints, (f) the wireframes in different stages of deformation from an ellipsoid to a dinosaur.

As future work, we intend to:

- Apply this technique to video sequences from different viewpoints and generate animation without traditional dynamic simulation;
- Design more complicated mesh consistency metric to handle specular surfaces and recover geometry and reflectance map at the same time.

References

[1] P. Fua, "A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features" *Machine Vision and Applications*, 6(1), 1993.

[2] P. Fua, Y. G. LeClerc "Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading" *International Journal of Computer Vision*, 16, 35-56, 1995.

[3] O. Faugeras, R. Keriven "Complete Dense Stereovision Using Level Set Methods" *Proceedings of the 5th European Conference on Computer Vision*, 1, 379-393, 1998.

[4] S. M. Seitz, C. R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring" *International Journal of Computer Vision*, 35(2), 1999.

[5] K. N. Kutulakos, S. M. Seitz, "A Theory of Shape by Space Carving" *Proceedings of the 7th International Conference on Computer Vision*, 307-314, 1999.

[6] S. Roy, I. J. Cox, "A Maximum-Flow Formulation of the N-camera Stereo Correspondence Problem" *Sixth International Conference on Computer Vision*, 492-499, 1999.

[7] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Surface Reconstruction from Unorganized Points" *SIGGRAPH'92 Proceedings*, 71-78, 1992.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, "Mesh Optimization" *SIGGRAPH'93 Proceedings*, 19-26, 1993.

[9] P. E. Debevec, C. J. Taylor, J. Malik, "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach" *SIGGRAPH'96 Proceedings*, 11-20, 1996.

[10] S. M. Seitz, C. R. Dyer "View Morphing" *SIGGRAPH'96 Proceedings*, 21-30, 1996.

[11] M. Levoy, P. Hanrahan "Light Field Rendering" *SIGGRAPH'96 Proceedings*, 31-42, 1996.

[12] S. J. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen "The Lumigraph" *SIGGRAPH'96 Proceedings*, 43-54, 1996.

[13] R. Szeliski, H.-Y. Shum. "Creating full view panoramic image mosaics and texture-mapped models" *SIGGRAPH'97 Proceedings*, 251-258, 1997.

[14] M. Garland, P. Heckbert, "Surface Simplification Using Quadric Error Metrics" *SIGGRAPH'97 Proceedings*, 209-216, 1997.

[15] P. Heckert, M. Garland, "Survey of Polygonal Surface Simplification Algorithms" *SIGGRAPH'97 Course Notes*, 1997

[16] L. Guibas, J. Stolfi, "Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams" *ACM Transactions on Graphics*, 4(2):75-123, 1985.

[17] S. Osher, J. Sethian, "Fronts Propagating with Curvature Dependent Speed: Algorithms based on the Hamilton-Jacobi Formulation" *Journal of Computational Physics*, 79:12-49, 1988.

[18] J. A. Sethian, *Level Set Methods*, Cambridge University Press, 1996.

[19] W. H. Press, B. P. Flannery, S. A. Teukolsky, W. T. Vetterling *Numerical Recipes in C*, Cambrige University Press, 1988.