Blinking Cubes: A Method for Polygon-Based Scene Reconstruction

by

John Andrew Harvey

Submitted to the Department of Electrical Engineering and Computer Science in Partial Fulfillment of the Requirements for the Degrees of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING
AND
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 22, 1998
[June 1998]

Author ______________________________
Department of Electrical Engineering and Computer Science
May 22, 1998

Certified by [illegible]
Leonard McMillan
Associate Professor of Computer Science and Engineering
Thesis Supervisor

Accepted by [illegible]
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Blinking Cubes: A Method for Polygon-Based Scene Reconstruction

by

John Andrew Harvey

Submitted to the
Department of Electrical Engineering and Computer Science on

May 22, 1998

In partial fulfillment of the requirements for the degrees of
Bachelor of Science in Computer Science and Engineering and Master of
Engineering in Electrical Engineering and Computer Science

Abstract

This thesis introduces a new approach to scene reconstruction that produces robust reconstructions of three-dimensional scenes from arbitrary sets of images and their corresponding camera calibration information suitable for viewing from arbitrary viewpoints. By iterating through a binary occupancy volume in the three-dimensional space of interest, the scene is reconstructed by posing a hypothesis of the scene structure surrounding each voxel based on the Marching Cubes algorithm for smooth surface reconstruction of volumetric data. This technique has the advantages of being able to handle significant occlusions and widely varying input viewpoints, while still producing a standard geometric description of the scene.

Thesis Supervisor: Leonard McMillan

Title: Associate Professor of Computer Science and Engineering

VI-A Program Thesis Supervisor: Thomas Malzbender

Title: Project Manager, Graphics Algorithms and Architectures Group, Hewlett-Packard Research Laboratories

Table of Contents

List of Figures

1 Introduction

Scene reconstruction is a difficult problem that has been studied for many years. Although the problem has traditionally been considered to be solely in the domain of machine vision, researchers in computer graphics and photogrammetry have also proposed various solutions. Despite this considerable amount of interest, the ability to robustly reconstruct accurate scene descriptions without the use of specialized acquisition hardware remains an open topic.

In this thesis work, I develop a new scene reconstruction algorithm that allows robust reconstructions of real and synthetic calibrated imagery and produces a triangle-based geometric approximation of the scene suitable for modification and viewing with standard graphics pipeline implementations.

There are many applications for scene reconstruction. In robots, reconstructions are used to analyze and identify the location, orientation, and shape of objects in a scene. With the recent growth of the Internet and electronic commerce, reconstructions of products can allow the creation of three-dimensional catalogs, where customers can get a complete view of products. In recent years, the concept of tele-presence [17] has been coined to describe the use of reconstructions to simulate the experience of being in a remote location. Scene reconstruction can also allow a smoother and more efficient integration of real-world and synthetic imagery.

Perhaps the most compelling motivation for scene reconstruction is the time required to produce a synthetic model of a real-world object by hand. Despite improvements in animation and modeling software, the manual production of detailed and accurate models remains a time-consuming, labor-intensive task that requires considerable skill on the part of the designer. In addition, previous methods have all addressed various approaches for reproducing representations of a scene. However, for purposes of integrating these reconstructions with other scene representations (real or synthetic), most of the methods cannot simply be inserted into a larger scene as a component. A major motivation for this work is the ability to build a representation of a real world scene or object that can be easily integrated into existing graphics authoring applications (i.e. special effects, architectural designs, etc.) and viewed through existing graphics hardware.

1.1 Approaches

1.1.1 Stereo Correspondence

One of the most widely used reconstruction methods is the recovery of depth by finding correspondences, or projections of the same feature, in images taken from slightly different locations. A pixel in one image is matched with pixels in another along a line called the *epipolar line* that is determined by the original pixel location and the viewing parameters. The pixel is matched by searching for a pixel whose difference in color from the original pixel is below some error threshold. Once a corresponding

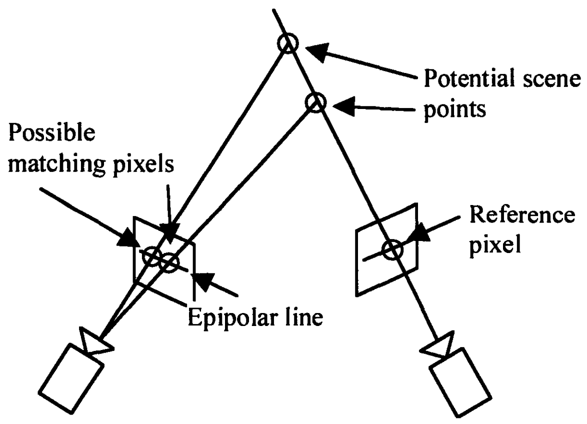


Figure 1: Stereo correspondence

pixel is found, the two pixels can be projected back into three dimensions, and the intersection of their projected rays is considered the location of the point. This process is repeated for each pixel, producing a depth map. A synthetic view can be rendered using this depth map by considering the points as vertices of a mesh of polygons and re-rendering these polygons or by reprojecting the actual pixels into the synthetic viewpoint.

Stereo correspondence provides a relatively simple method for recovering depth maps of images. However, it has significant disadvantages that make it unfeasible for many reconstruction applications. First, the depth maps use no information about how objects block the viewing of other objects. For example, if there are two points, A and B in a scene, both could be visible in a first image, but A could be hiding, or *occluding*, B in another, second image. If the first image is used as the reference, it would search for pixel(s) corresponding to point B in the second image, even though point B is occluded from the second image. The search will either fail or find an incorrect match. In general, in the presence of significantly different occlusion relationships, many pixels will find a false correspondence or not find a correspondence at all. False correspondences also arise when the images contain regions of uniform color or periodic structures.

Stereo correspondence also has trouble if a surface is viewed from two very different angles. The area a rectangular surface projects to in an image when it is parallel to the image plane is much greater than the area when the surface is nearly perpendicular to the image plane. The rectangle is significantly *foreshortened* in the image when it nearly perpendicular. Stereo correspondence ignored the effects of foreshortening and assumes that all surfaces are parallel to the image planes when making correspondence searches, which will lead to inaccuracy for surfaces viewed.

One approach to mitigating both the occlusion and foreshortening problems is to constrain correspondences to be between images that are very close to each other. Keeping the images close decreases the difference between views and thus the likelihood that a point will be visible in one image and occluded in another. Keeping the images close also decreases the likelihood that the amount of foreshortening will have changed significantly. This approach has two disadvantages. First, the similarity of the views means that the method becomes very sensitive to noise in the image measurements. Second, keeping images very close makes it difficult to obtain input for a large-scale environment.

1.1.2 View Synthesis

Recent research in scene reconstruction has examined a subset of the scene reconstruction problem: the construction of images for synthetic viewpoints, or image-based rendering. Instead of computing a global model suitable for viewing, image-based rendering generates synthetic views by using two-dimensional input images to produce new two-dimensional images that correspond to what would be seen from synthetic viewpoints. This can be considered to be a shift from examining the structure of the scene to examining the views of the structure of the scene. The structure of the scene is never explicitly determined.

While these methods show promise for navigation of a complete scene, the lack of a global model results in highly inaccurate integration of image-based scenes with other image-based scenes or synthetic objects. Image-based scenes have the complete lighting solution for the particular configuration in which they were acquired. This is adequate for displaying the scene as it was captured, but the application of new lighting models, including reflection between objects in a single image-based scene or a composite scene, would not produce an accurate solution. In addition, these methods tend to be very inefficient in storing scene information, since storing multiple views of the same scene results in significant amounts of redundant information.

1.1.3 Structure from Motion

Another method for reconstruction is to use a sequence of images taken from two (or more) viewpoints. By manually specifying pairs (or tuples) of points in the images that correspond to the same three-dimensional point, it is possible to determine not only the location of the points in space but the position and orientation of the viewpoints as well. This technique has been thoroughly investigated, and improvements to the original technique have included the use of more than two images and multi-pass solutions. While this technique is useful for camera calibration, it has limited utility as a reconstruction method. Depth is calculated by the manual specification of matching points, which is not feasible for high-resolution imagery. Similar to stereo correspondence, structure from motion is highly sensitive to sensor noise when image viewpoints are very similar. Finally, since the correspondences are normally specified as points (or edges), the resulting model is not a true three-dimensional model. A post process is required if the results are to be used as a model of the scene.

1.1.4 Specialized Acquisition Hardware

There has also been interest in determining depth as part of the acquisition process ([3],[12]). This process is the goal of *range-imaging sensors* that use various techniques to sample depth. Some of the more prevalent types of sensors are radar-based sensors, active triangulation sensors, and lens focusing sensors. Radar-based sensors recover depth by sending a signal (continuous or pulse) and measuring the time of flight or change in amplitude or frequency of the signal. Active triangulation scanners use a light projector and a camera to recover depth. The projector emits some type of structured light (point, line, grid of points, etc.) onto the point or surface in question, and the position where the reflected light projects onto the camera plane is captured. The location of the point or surface can be recovered by measuring the angle and baseline distance between and the position of the light projector and the projection on the camera plane. Lens focusing recovers depth by moving a lens along an axis until the light reflected off of a point through the lens projects to a minimum size (the point is then assumed to be in focus). Depth can then be recovered with knowledge of the focal length of the lens.

While depth acquisition hardware would seem to eliminate the need for other reconstruction methods, there are drawbacks that prevent it from being a complete solution. First, range-imaging sensors collect a relatively sparse set of points. This means that if the reconstruction is used in applications requiring a surface model, and especially a closed surface, there must be an additional method for generating a surface model. Furthermore, the scanners are designed to examine single objects or small scenes and, as a result, are not well suited for capturing large-scale scenes, especially those outdoors.

Finally, because many of the scanners use the observation of reflected light as the means for determining depth, the results are affected by highly specular or very dark surfaces [3].

1.2 Issues

There are a number of important issues involved in reconstructing a scene. First, consider that objects in the scene can occlude each other. This means that objects visible in one image can be partially or completely obscured in other images. Second, the shading of objects is also an important issue. Many materials reflect light differently based on viewing direction. This property causes a particular surface to appear different from each viewpoint. Most reconstruction algorithms have simplified this issue by assuming a scene full of diffuse objects. Third, the cameras and their corresponding images must be calibrated to allow accurate mapping of image points back into three-dimensional space and vice-versa. Finally, if a reconstruction is to be accurate from a wide range of viewpoints, a reconstruction algorithm must accept a wide breadth of input images.

There are a variety of representations for scenes, and each has certain advantages and disadvantages. Some current representations include polygons, volumes of three-dimensional points (voxels), and parameterized surfaces (Bezier curves, NURBS, etc.). Using a polygonal representation means that existing hardware graphics pipeline implementations can be used for fast rendering of reconstructions. In addition, with the pervasiveness of polygon, and in particular triangle-based environments for rendering, producing a polygonal representation greatly simplifies the task of integrating reconstructions into existing three-dimensional models and applications.

2 Previous Work

Blinking Cubes falls into a special class of reconstruction algorithms known as image-based scene modeling, which is scene recovery for purposes of subsequent viewing or manipulation. Work in image-based scene modeling can be roughly partitioned into two general approaches. Papers from each approach will be discussed, noting the advantages and limitations of each. It is also important to note that it will be assumed that accurate camera calibration can be suitably achieved, and for this reason the various methods for acquisition of imagery will not be discussed.

The first approach focuses primarily on methods for producing a representation of the scene that allows viewing from arbitrary viewpoints, with or without an underlying physical description of the scene. The second approach is the reconstruction of scene structure from images. In this case, the goal is to produce a representation, or subset thereof, that contains accurate information about the location of attributes in the scene.

2.1 View Synthesis

2.1.1 Chen and Williams

One of the first attempts at image-based rendering was the view interpolation work of Chen and Williams [2]. Using previous knowledge of camera calibration and pixel depth, this method computes a high-density "morph map" that describes the flow of pixels from one image to the next. These images and maps form a graph where the nodes are the images and the maps are the edges. Synthetic views are generated by interpolating between the motion vectors from the two closest images. Since the mapping of pixels can be many-to-one or vice-versa, this technique compensates for overlapping regions with depth buffering and for holes by interpolating from neighboring pixels. One of the strengths of this technique, and of all image-based techniques, is that the rendering time is dependent on the image resolution and not on the scene complexity. This is a significant gain for images generated by complicated shading models and/or with scene models of significant complexity.

View interpolation is a viable method for computing intermediate views of a pre-established scene. However, the lack of any actual scene representation means that it is a technique limited to viewing only. In addition, this method relies on pre-computed accurate depth maps, which are currently only robustly available from synthetic imagery.

2.1.2 View Morphing

One of the most visible techniques for view synthesis has been morphing. Its most popular use has been in music videos and movie special effects to transition between two different images of the same type of object or between different views of the same object. One of the drawbacks of the traditional approach was the need to manually specify structure points in the input images and their movement through the morph sequence. View morphing [15] simplifies this process by performing a pre- and post-process to prevent physically implausible distortions to imagery and produce plausible intermediate views. This process, known as rectification, projects the reference images into parallel viewpoints. From these rectified images, rectified intermediate images are produced through standard interpolation techniques. The final image is produced by applying an inverse rectification operation to adjust the camera viewing direction back to the proper interpolated view.

View morphing is an efficient method for generating intermediate views between two images. While it is a simplification, it still requires the manual specification of tie-points in order to efficiently produce intermediate frames. In addition, since the morph is moving between two reference frames, only viewpoints that lie along the line connecting these two frames will produce valid images. Also, since occlusion relationships are assumed to be unchanged by the morph, any changes in the visibility order will cause significant artifacts in artificial views.

2.1.3 Light Field Rendering

Similar to [2], Light Field Rendering [9] is an interpolation method for computing synthetic views. A light field is a regular sampling of the viewpoints for an object or scene, which obtains a large number of closely spaced samples. A four-dimensional representation of these samples is used in which two pairs of two parameters each represent points on two different parallel planes. These four parameters denote a ray in space which intersects a point in the three dimensional space of interest. Synthetic views are generated by projecting the viewing ray and computing its intersection with the aforementioned planes. From this, pixel color is a straightforward lookup in the four dimensional light field. These lookups form a set of discrete color samples that are used to color pixels in the synthetic view with various types of interpolation.

As expected, this operation has a very low per-pixel rendering cost. However, the boost in efficiency is countered by the large amounts of data that are required for a light field to avoid excessive blurring at synthetic viewpoints. In addition, since no three-dimensional structure of any kind is recovered, there is no existing method to integrate multiple light fields in a composite view.

2.1.4 The Lumigraph

The Lumigraph [5] is based on a similar representation to [9] with some additional steps in reconstruction to accommodate variations in geometric structure. Whereas light fields were made from both object- and scene-centric approaches, the Lumigraph is more focused on the acquisition and representation of images of an object, primarily for the construction of image-based object primitives. The construction of a particular Lumigraph is similar to that of a light field, but there is an additional emphasis on sampling issues. A blue screen technique is used in acquisition to differentiate between foreground and background, allowing the construction of an approximation of the object's convex visual hull. This approximate geometry is used to refine the Lumigraph representation to more accurately reflect the true object shape and to assist in the scaling of imagery resulting from the Lumigraph. Imagery can be produced via ray tracing or texture mapping, again using the approximate geometry for depth correction.

Since the Lumigraph uses a very similar representation to that of light fields, it also requires a considerable amount of storage space for a sample. In addition, the use of silhouettes to construct a geometric representation results in significant inaccuracy for objects containing a high degree of concavity. This rough approximation also causes inaccuracy if multiple lumigraphs are to be used together or inserted into a scene.

2.1.5 Virtualized Reality

Kanade et al. [7] use reconstructions to allow the viewing of videotaped sequences from synthetic viewpoints. Using a multi-baseline stereo method similar to [8], they acquire images of a scene located inside a geodesic dome with video cameras at each vertex. After synchronizing the images in time, a 2 1/2-dimensional model is reconstructed for each camera frame at each instant. Using a neighborhood of the closest cameras, stereo correspondence search is run along the epipolar lines in the neighboring camera frames. This produces a dense depth map for each frame of each camera. From these depth-enhanced images, synthetic views are generated by creating a triangle mesh from the depth points and re-rendering the nearest reference image into the new viewpoint. Neighboring cameras, called supporting cameras, are used to resolve areas of ambiguity left by the reference image.

Virtualized Reality is able to reconstruct areas of reasonably high texture due to its high number of uniformly spaced samples. Using multiple images to find correspondences produces a dense, high-quality estimate of depth for each image. However, these depth estimates are not integrated into a complete whole, and as a result, there is no actual model used to produce the new viewpoints. In addition, these depth maps require manual editing to produce results suitable for subsequent use. Finally, the lack of global integration means that inconsistencies between the sensors for neighboring cameras can cause artifacts to arise in synthetic views.

2.1.6 Plenoptic Modeling

McMillan and Bishop [11] construct cylindrical panoramas from a sequence of images recorded by a rotated camera. From these panoramas, mappings between pixels in one projection to pixels in another are specified using stereo correspondence with a cylindrical epipolar constraint, which produces the same reduction of correspondence search from two dimensions into one that epipolar lines do for planar projections. From this, a dense image flow field is computed and used to compute the projections for intermediate viewpoints located between two reference viewpoints. In the case of two points mapping to the same point in a synthetic view, the points are drawn in a raster order such that the point closest to the synthetic view is drawn last. This guarantees that the proper point is always drawn last and is thus visible. Unfilled pixels are interpolated using methods similar to [2].

Plenoptic Modeling is able to produce images at synthetic viewpoints in real-time with proper occlusion and perspective effects. This is beneficial for navigation of a real-world scene, but, because the synthetic views are produced by resampling the original input pixels, only viewpoints that are close to reference viewpoints produce results that are of similar quality to the input views. Additionally, since the mappings between viewpoints are obtained by the stereo correspondence approach, Plenoptic Modeling is subject to the artifacts that affect all stereo correspondence methods.

2.2 Structure from Images

2.2.1 Feature Tracking

Seitz and Dyer [13] use a method that guarantees an optimal reconstruction of every continuously visible scene feature in a set of images. A preprocess is used which detects image features (points, lines). From the list of features for a particular image, one feature is chosen, and an attempt is made to reconstruct the feature by corresponding it across all of the images. This process is then repeated until the list of new features is exhausted.

This method is good for robust extraction of scene features. However, it is limited to tracking simple features of a scene, and, as a result, without considerable manual effort, the recovered description will always be incomplete. In addition, requiring that any detectable feature appear in every image means that for scenes with significant occlusion, few, if any features can be reconstructed.

2.2.2 Kang and Szeliski

Similar to [11], Kang and Szeliski [8] use cylindrical projections to construct scene descriptions. However, they use the projections as an intermediate step in producing a mesh of three-dimensional points. The points are recovered using various techniques that are able to correspond points from a small number of cylindrical panoramas.

Although this method is comparatively fast method for generating a reconstruction, it is limited by the use of a small number of cylindrical panoramas. It is advantageous in the case of reconstructing complete rooms for viewing or navigation. However, in addition to limiting the vertical field of view, input images are concentrated at a handful of viewpoints. While the range of viewing angles from each of these locations is complete, it becomes limited in situations where it is desirable to examine particular objects within a scene.

2.2.3 Architecture from Photographs

Debevec et al. [4] have produced a hybrid method for producing a geometry- and image-based representation of architectural scenes. Noting that traditional stereo correspondence places cumbersome constraints on acquiring information about large, outdoor scenes, the hybrid approach seeks to make modeling a computer assisted, human driven process, and refinement and viewing a computer driven process. Models are constructed through a program that uses parameterized blocks as its primitive. Users specify constraints on blocks to produce an approximate model of the architecture of interest. They then mark points and edges in the input images (normally photographs) that correspond to model features and the computer refines the model to optimize the correlation between the images and the model.

Once the approximate model of the architecture has been recovered, a technique known as model-based stereopsis is used to refine the model to account for unmodeled detail and to compute depth maps for the input images. Model-based stereopsis correctly computes correspondences from images in widely varying viewpoints by projecting the second, or offset image onto the approximate model and then into the image plane of the first, or key image. This helps to reduce the effect of different foreshortening patterns, which is a significant problem in traditional stereo correspondence. Correspondences are then searched along the first image's epipolar line in the image of the warped second image. Once depth maps have been computed for each of the input images, re-rendering can be accomplished with standard image-based rendering methods ([2], [11]).

The architecture from photographs method for scene reconstruction makes significant advances in addressing the shortcomings of stereo correspondence. The use of an approximate geometry greatly simplifies the task of depth map recovery. However, this approximate geometry must still be created manually, and if a highly detailed model is to be recovered, a corresponding amount of effort must still go into the model. In addition, since the modeling program is geared toward architectural scenes, it is not well suited as a general-purpose reconstruction method.

2.3 Voxel Coloring

Seitz and Dyer [14] explore a technique known as voxel coloring that reproduces a volumetric representation of a scene given a set of images and camera calibration information. This technique has also been used to allow two-dimensional image editing to propagate across multiple views of a scene [16]. From the reconstructed volume, views of the scene can be produced that reproduce the original input images and allow viewing from synthetic viewpoints through standard volume rendering techniques. This technique is something of a hybrid between model recovery and view synthesis, and one of its most interesting aspects is that it is able to compute depth estimates and recover scene structure in unison, two operations that are typically done sequentially. Despite the fact that it constructs a global model that is viewable from any direction, the model is constructed by focusing not on reproducing accurate scene structure but on maximizing the correspondence between input images and the reconstructed scene as it is viewed from the input viewpoints.

Voxel coloring operates by defining an initial volume shape and dimension (e.g. a cube) that will contain the scene of interest. In a pre-process, foreground and background can be differentiated to avoid the unnecessary reconstruction of background structure. The coloring then proceeds iteratively by testing each voxel for a correspondence. The scanning order of voxels is constrained to move away from the convex hull formed by the input camera locations. This follows from a defined ***ordinal visibility constraint*** which makes the assertion that by moving away from the camera hull, occlusion can automatically be accounted for by masking out pixels when they are used to color a voxel. As each voxel is visited, it is projected back into each input image to form a set of pixels that will be examined for correspondence. Pixels that have been masked are not taken into consideration. The sum-of-squares error in this occlusion-reduced set of pixels is computed, and if it is below a certain threshold, the voxel is marked as colored with the average pixel color, and the pixels used in the correlation are marked as masked. This technique continues until all voxels have been examined, resulting in a dense volumetric approximation of the scene.

The voxel coloring technique is efficient and robust for multiple reasons. First, it does not require user intervention in the form of manual correspondence matching. Second, because the technique iterates by moving through voxels in the volume instead of pixels in the input images, there is no need to specify a reference image. Additionally, since there is no use of epipolar lines or similar correspondence based approaches, the technique can handle scenes with severe occlusion and/or significantly different viewpoints between images. Finally, since each voxel in the scene volume is colored independently, only the input images, the masks used to denote whether a pixel has already been associated with a voxel, and the current voxel of interest must reside in memory at one time.

One disadvantage to this approach is the ordinal visibility constraint. While some useful configurations obey this constraint, there are many situations in which allowing the input cameras to violate the constraint would be advantageous or even necessary. The visibility constraint also makes no specification for iteration through the minor axes (i.e. in each "slice" of the volume). This ambiguity means that the assertion of a complete ordering to account for occlusion is not entirely correct. In addition, in cases of significant scene complexity, the voxel representation can require a considerable amount of storage. Because correspondences are computed across multiple images without regard to illumination, voxel coloring is also limited to stationary scenes with approximately Lambertian surfaces.

Similar to stereo correspondence methods, voxel coloring has difficulty with areas of uniform brightness. In this case, erroneous voxels will be colored that do not correspond to actual scene points. These erroneous voxels are described as *cusps*. Since the reconstruction moves from near to far, these uniform areas result in voxels that are colored closer to the camera volume than the actual scene structure that their pixels correspond to. This results in a bias, or cusping, of the reconstruction toward the cameras (see Figure 2). It is important to note that while cusps may be inconsistent with the scene structure, they are visually consistent with the input images, and they will in fact correctly reproduce the input images. Thus, from the perspective of the voxel coloring algorithm, the cusp geometry is

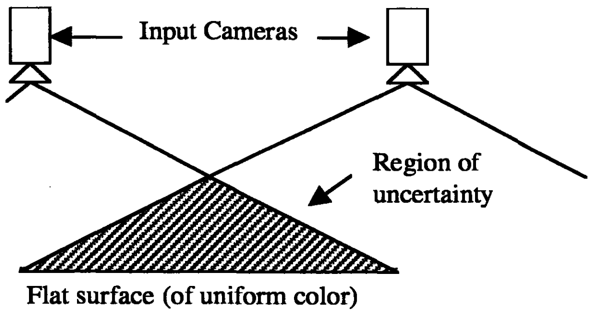


Figure 2: Cusping in surface reconstruction

just as “correct” as the actual geometry. Cusping is the result of ambiguity in the input, and while it can be mitigated by introducing more images, it cannot be eliminated.

3 Framework

3.1 Issues in Scene Reconstruction

3.1.1 Occlusion

For a scene reconstruction technique to be widely applicable, it must have a method for dealing with changing occlusion patterns in images. As the viewpoint of a scene changes, surfaces appear and disappear in relation to the structure of the scene. If a reconstruction method does not account for these changes, many inaccuracies arise. Correlations will fail and true points in space will be missed because the point will incorrectly project to images that cannot see the point.

3.1.2 False correspondence

Likewise, it is important to address the issue of correspondence between pixels of similar color which represent different points in space. While this event is statistically unlikely in the case of scenes with few areas of similar color, it is of considerable interest for flat surfaces of uniform or near uniform color. Error thresholds in reconstruction algorithms must be high enough to allow for sensor noise and/or sampling inaccuracies. This leads to pixel regions of effectively uniform color, making any kind of correlation search strategy prone to false matches in these regions. Without a previously established estimation of scene structure, there is no automated method for avoiding these false matches.

3.1.3 Shading

Another important issue when addressing reconstruction accuracy is surface reflectance. If a surface's reflectance is a function of the viewing direction, simple methods for determining a correspondence will be unable to match pixels across images. This has traditionally been accounted for by assuming a scene full of diffuse or near diffuse surfaces, but if a method is ever to be considered completely robust, it must be able to account for these variations in reflectance.

3.1.4 Confidence

Another important issue is the global utility, or confidence in the accuracy of a reconstruction. Many of the previously examined techniques discuss the importance of having a wide range of viewpoints in order to produce a quality reconstruction. In general, the problem of guaranteeing a complete reconstruction is ill posed. For any set of input viewpoints of a scene, it is always possible to construct a scene in which a portion of the structure is not viewed by any of the input viewpoints. For some reconstructions, this is not a significant issue, because their application does not require a globally consistent result. However, if a method's goal is to produce a general-purpose representation of the scene or object, it must be able to acquire multiple images from many different viewpoints in order to obtain a sufficiently representative sampling of views of the object.

3.1.5 Ordering of Iteration

Another important issue to address is the order that an algorithm examines a three-dimensional space in order to produce a reconstruction. This issue is complicated because determining color is affected by previously determined occlusion relationships, which are likewise determined by previously determined colors. In order for an algorithm to be considered deterministic, it should be possible to determine a non-arbitrary order through which the space is traversed.

3.1.6 Accuracy

Perhaps the most important issue in examining a scene reconstruction method is the accuracy of the results it produces. *If a reconstruction method is to be considered a valid approach, it must be able to produce a representation that can reproduce the input views and a range of synthetic views with a high degree of plausibility.* If a reconstruction is unable to produce the input data with a reasonable degree of accuracy, it can hardly be considered a viable technique.

The traditional method for evaluating accuracy has been some kind of per pixel comparison between acquired imagery and view generated from the reconstruction through the same viewpoint. While this is probably not the best metric possible, it is the most widely accepted. For this reason, I will make the simplifying assumption that this is our primary measure of accuracy in reconstruction.

3.2 Design

I propose a new method for scene reconstruction, called Blinking Cubes, based on the voxel coloring approach of Seitz and Dyer. The algorithm produces a polygon-based model that maximizes the correspondence between its input images and renderings into the input viewpoints.

3.2.1 Marching Cubes

Marching Cubes [10] is a technique for constructing surface models from point volumetric data. Originally intended for use in making volumetric medical data more easily understood, it is a technique for fast local analysis of voxel density to produce geometry that approximates the surface around the voxel. The data structure examined is a volume of binary voxels that represent the presence or absence a previously classified material or surface. Each voxel is associated with a point in a cube lattice. Since there are eight voxels associated with each cube and each voxel can have two different states, there are 256 different possible states for each cube in the lattice (exploiting symmetry reduces the number of states to 15).

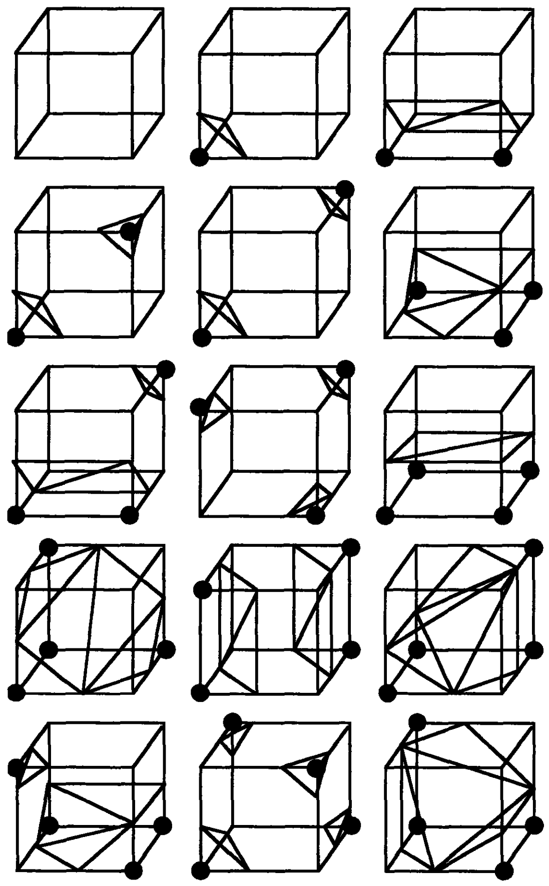


Figure 3: Geometry look-up table for Marching Cubes. Diagram based on figure in [10].

By a pre-process analysis of the way in which a surface would give rise to a particular configuration of occupied and empty voxels, each of these possibilities is paired with a set of triangles whose vertices lie on the edges of the cube. These configurations are usually stored in a look-up table (see Figure 3). The triangles are used to form tessellated surface(s) within the cube. Because adjacent cubes use overlapping voxels and edges, and by virtue of the geometry associated with the indices, surfaces are guaranteed to be closed and connected as long as they have continuous adjacency in the voxel representation.

3.2.2 Generating Geometry with Marching Cubes

Blinking Cubes uses the geometry table (described in Section 3.2.1) from Marching Cubes to generate potential surfaces for reconstruction. As each new voxel is visited, it is marked occupied, causing a change in the geometry of the current voxel and the eight surrounding voxels that use the current voxel as part of their Marching Cubes lookup (see Figure 4).

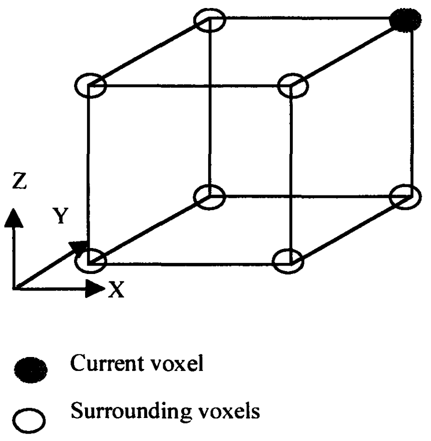


Figure 4: A voxel and its neighborhood

In the case of voxels that already have existing geometry, the old geometry and its corresponding projection(s) must be saved and labeled as old before new geometry may be generated. Note that the geometry generated by the current voxels and its eight neighbors is considered atomic for purposes of introduction into the permanent reconstruction model, but that each triangle is separately correlated with the input images.

3.2.3 Occlusion and Depth Ordering

As voxel coloring reconstructs voxels, each voxel's color is derived from the set of pixels that it projects to in the set of input images. Once a reconstruction has been validated, these pixels are masked out and ignored in further correspondence calculation. However, it is possible that subsequently reconstructed voxels will project to the same pixel location(s). This brings rise to an important issue: How is the visible set of pixels determined for a geometric face? Voxel coloring makes the assumption that once a pixel has been used in a successful coloring of a voxel, any subsequent voxels should ignore the pixel in their correspondence calculation. This assumption works for voxels located in different slices of the reconstruction volume. However, in the case where voxels, or in the case of Blinking Cubes, faces, are located in the same slice, this can lead to ambiguity.

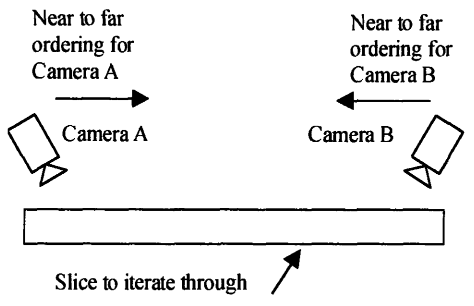

Near to far ordering for Camera A
Near to far ordering for Camera B
Camera A
Camera B
Slice to iterate through

Figure 5: Ambiguity in minor axes iteration

Consider the case in Figure 5: It is not possible to iterate through the slice and guarantee a near to far ordering for both cameras. Many camera configurations obeying the ordinal visibility constraint do not have an ordering that guarantees a monotonic increase in depth for all cameras. To compensate for this, an LDI (layered depth image) [6] representation is used which contains multiple values for each (x, y) location in the image. For each of these values, a camera space depth and a triangle identification number is stored. Color does not need to be stored here because each triangle has a constant color. These values are sorted in ascending order by depth. When a new set of geometry is rendered, all values that have triangle IDs corresponding to geometry that would be replaced is the correlation succeeds are temporarily taken out of the image representation. The new geometry is sent through the traditional graphics pipeline, one face at a time, a difference being that the color channel is used to store the triangle ID. The resulting depth and ID values are then inserted into the layered depth images. The current color of a pixel is computed by looking up the color of the triangle associated with the ID with the depth pixel value of the smallest depth.

The LDI representation allows each camera to do proper depth ordering by storing every visible point along a viewing ray. In the case of Figure 6, the slice is iterated from left to right. Camera B could reconstruct geometry early in the slice and then have subsequent geometry generated that intersects the same viewing ray(s) as the previous geometry in the slice. A single occlusion flag would have already marked the pixel as used from the first geometry and prevent Camera B from "seeing" the pixels used to color the previous geometry. This pixel is hidden from the new geometry, even though the new geometry is closer to Camera B. By using the LDI, Camera B will know that the later geometry is closer and correctly use pixels that project to both the previous and current geometry. This use of the LDI representation causes another difficulty. Assume that valid geometry is found at the second location. The first location has now been colored with pixels that it shouldn't be seeing from Camera B. Likewise, it is possible that some geometry at the first location failed because the geometry at the second location had not yet occluded its pixels.

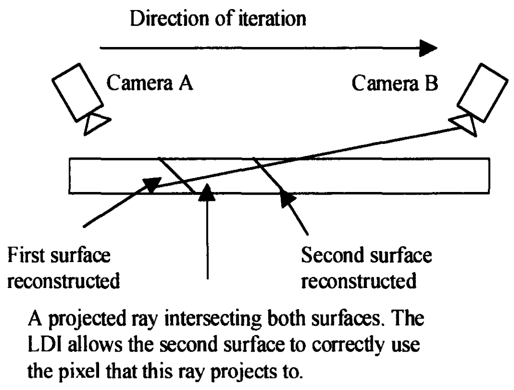

Direction of iteration
Camera A
Camera B
First surface reconstructed
Second surface reconstructed
A projected ray intersecting both surfaces. The LDI allows the second surface to correctly use the pixel that this ray projects to.

Figure 6: Using the layered depth image to prevent incorrect occlusion masking

This ambiguity is a symptom of the fact that the cameras have conflicting depth orderings. One approach to handling this problem is to use a subset of the input cameras that have unambiguous lines of sight to the geometry to be tested. Such an algorithm could proceed by choosing the geometry with the largest number of available cameras at each iteration until no proposed geometry is adequately visible. While this method could guarantee proper depth ordering, it diminishes the advantage of having many views available and thus decreases the confidence with which correlations are made.

3.3 Method

The proposed method is as follows:

An input set of calibrated images is provided along with initial location and dimensions of a binary volume (all voxels are initially marked as empty). The volume is iterated in the same near to far manner as [14], except that a new method is used to determine ordering within each two dimensional slice of the volume (described in 3.3.2).

3.3.1 Correlating a Voxel

A voxel correlation is attempted by marking the current voxel as occupied. With this new voxel occupancy configuration, a new set of potential scene geometry is generated (as described in Section 3.2.2). This new set of geometry is projected back into the input images and a correlation similar to voxel coloring is attempted, with one exception. Because triangles are being generated instead of voxels, an operation to reduce the camera set is run by eliminating from consideration those cameras that cannot see the front side of the triangle (i.e. back face culling). Note that the LDI points from the current voxel neighborhood's geometry are temporarily ignored.

If the geometry passes the correlation test, the triangles are colored with the average of their projected pixels' colors. Thus, at each voxel, the occupancy causes new geometry to blink into existence and, if the changes are not satisfactory, blink back out of existence.

3.3.2 Iterating Through a Slice

As previously discussed in Section 3.2.3, a set of cameras obeying the ordinal visibility constraint can still have ambiguous depth ordering within the minor axes of a reconstruction volume. For a set of cameras to have uniformly ordered depth, all of the cameras must be located in the same octant above the reconstruction volume (see Figure 7). The use of camera configurations with a wide variety of views requires the violation of the depth ordering, at the very least in the minor axes and as a result, the best reconstructions would result from some type of global optimization approach such as tabu search or simulated annealing applied to each slice. However, the storage and speed requirements for such an approach currently make it computationally infeasible. Instead, the following greedy algorithm heuristic is used:

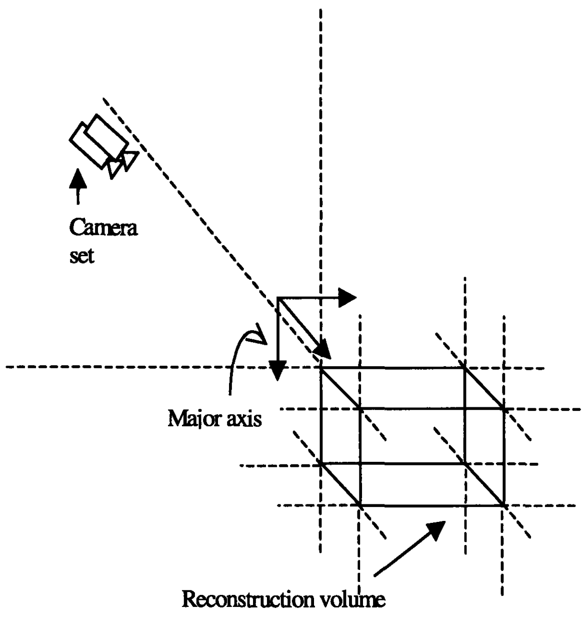


Figure 7: Camera set configuration to allow a uniform depth ordering for all cameras

The per voxel correlation method (described in Section 3.3.1) is run for each voxel in a slice, ignoring the correlation information resulting from other voxels in the same slice. Note that geometry from previous slices is taken into consideration (i.e. faces in the slice above affect faces in the current slice). Those voxels whose geometry pass the correlation test are ranked according to the quality of their correlation. The voxel whose set of geometry has the highest average correlation is committed and becomes part of the voxels that are taken into consideration. All voxels in the slice whose generation of geometry is affected by this committal are updated and the remaining, uncommitted voxels that have valid colorings are ranked again. This process of picking the best correlation, committing it, and updating the surrounding voxels is repeated until there are no voxels remaining with valid colorings. Note that the updating process can cause voxels that initially had valid colorings to become invalid and vice versa.

This process is repeated for each slice until all slices have been visited.

3.4 Advantages

Blinking Cubes has a number of benefits. The use of a polygon-based representation is advantageous for several reasons. Using faces instead of voxels introduces an automatic constraint that allows only those cameras that could all potentially see a face to be used in the face's correlation. In addition, objects in the real world are closed and connected, that is they do not have arbitrary holes or floating point artifacts produced in voxel reconstructions. By using the Marching Cubes algorithm, Blinking Cubes can guarantee that all surfaces in the resulting model will be closed. Blinking Cubes also benefits from the process of marking used, or masked, pixels. As in voxel coloring, this process automatically accounts for occlusion by removing from consideration pixels that have already been associated with a surface. However, by storing multiple points with the layered depth image representation, Blinking Cubes allows each image to store a more accurate representation of their occlusion history and prevent erroneously ignoring visible pixels.

One of the most significant advantages of Blinking Cubes is the greater accuracy of its correspondence approach. An important problem with stereo correspondence is the aligning of search windows to the image plane. This is an implicit assumption that the feature being examined always projects to the same area in both images, which is rarely the case. Quite simply, the effects of foreshortening sue to surface orientation are not considered. By iterating in a scene-centric fashion and generating possible matches in world space, this method automatically accounts for the differing projection areas and shapes corresponding to different viewpoints. Finally, since most computer graphics tools use polygons as one of the fundamental units of scene structure, the generation of a polygonal model means that the reconstruction can be viewed and modified with any of these tools.

3.5 Disadvantages

Blinking Cubes also has a number of disadvantages. Because it is based on the near to far iteration technique of voxel coloring, it is also susceptible to the cusping problem of surfaces being generated too close to the camera volume. Blinking Cubes also assumes accurate camera calibration. In the absence of accurate calibration, the quality of the reconstruction is likely to degrade quickly. In addition, if the scene is of considerable complexity and the voxel resolution is very high, the models reproduced can be very complicated. A considerable amount of storage space must also be available to store the input images and their associated layered depth images, which will probably have more than one color per point. However, this can be addressed using various surface simplification techniques. Blinking Cubes is also constrained by the ordinality visibility constraint of voxel coloring, which limits the possible image acquisition configurations. The lack of a completely well-defined ordering for iteration means inaccuracies can rise from incorrectly missed correlations and thereby reduce reconstruction accuracy. Finally, Blinking Cubes makes no efforts to account for non-Lambertian reflectance models. Correlation is computed blindly across different viewpoints, which are assumed to have a similar enough color to correspond if they are views of the same scene structure. This can cause a loss of detail in the reconstruction, or, in the case of surfaces with a significant amount of specularity, an inability to correctly rebuild the surface.

4 Results

The Blinking Cubes algorithm was implemented in C++ on a Hewlett-Packard HP9000-J282 workstation and tested both on HP and SGI workstations.

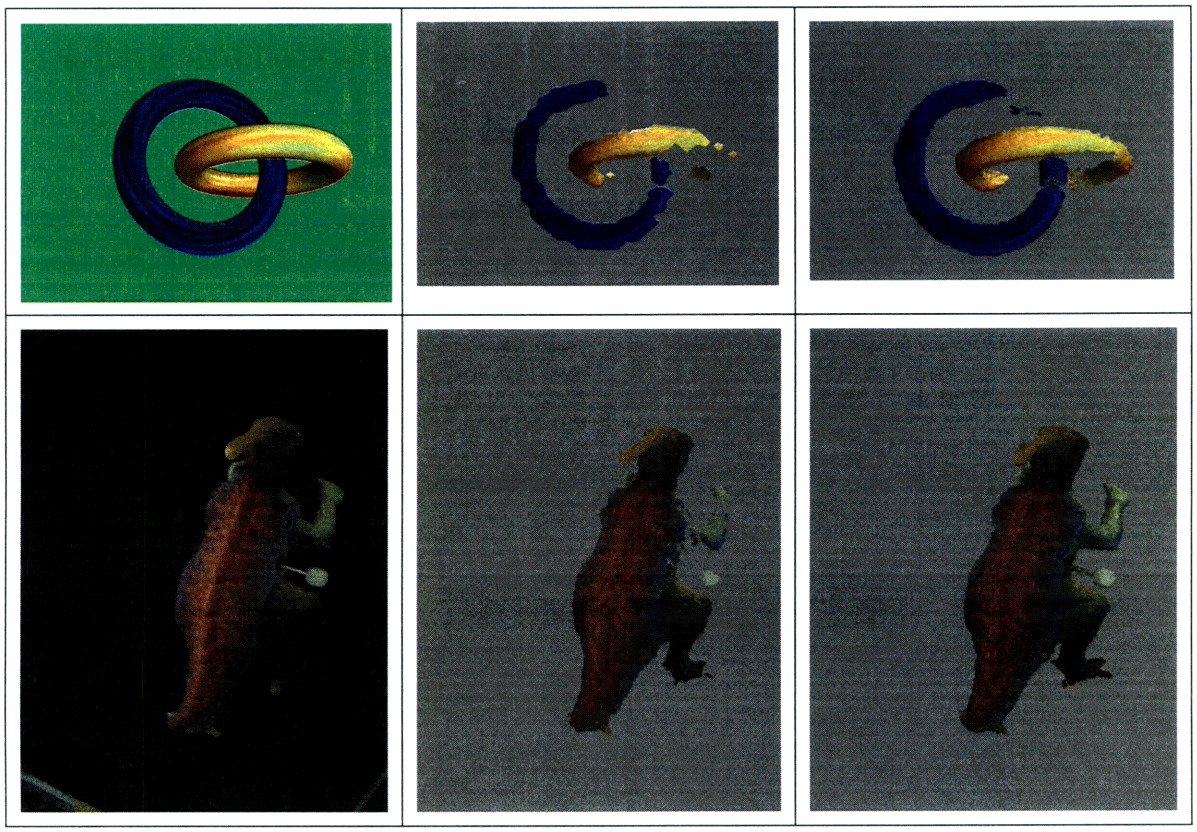


Figure 8: Effect of voxel resolution on reconstruction quality

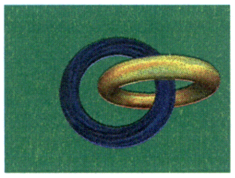
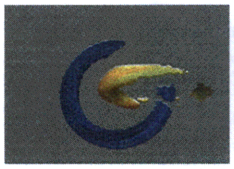
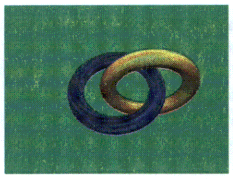
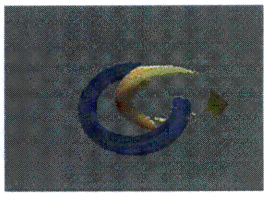
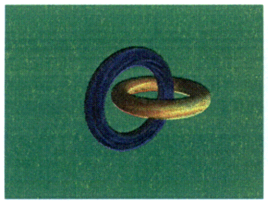
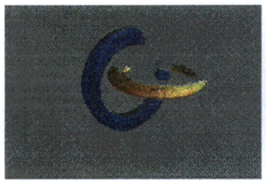
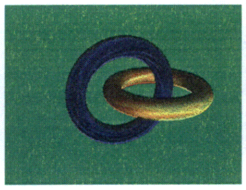
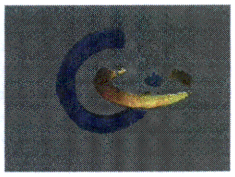
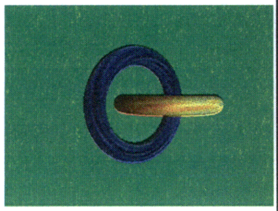
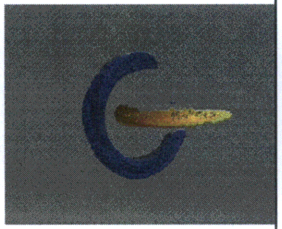
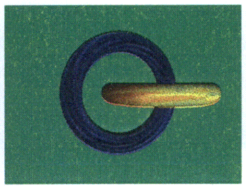
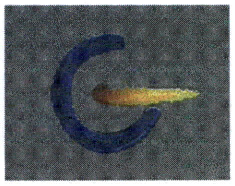
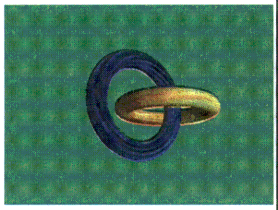
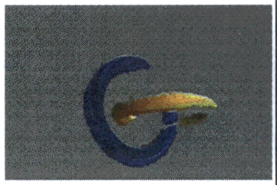
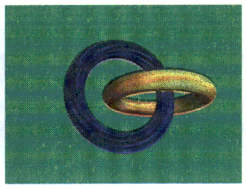
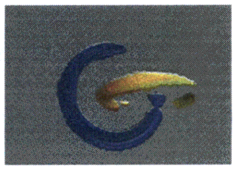
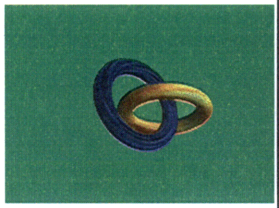
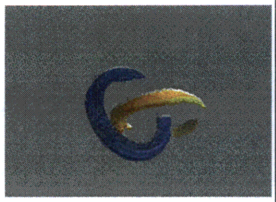
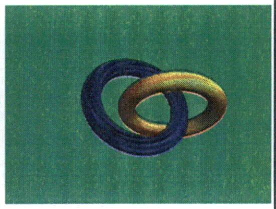
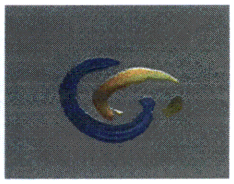
Tests were run with both real world and synthetic imagery at varying resolutions. The effects of varying resolution are shown in Figure 8, and Table 1 summarizes the results from reconstructions at two different resolutions for both data sets. The average percentage error indicates the average percentage difference in color between a pixel in the original input image and the corresponding pixel in the rendering of the reconstruction. Both percentage errors were computed for the higher resolution reconstruction. The running times of Blinking Cubes are longer than those of runs at similar resolutions in [14], but this is to be expected since testing each voxel consists of removing previous geometry and looking up new geometry for the eight voxels, rendering each of the new triangles, and inserting LDI points once the correlation has succeeded or failed. Note that the reconstruction images are often of different size than the input images due to an optimization that avoids the storage of unused input pixels.

Table 1: Results comparison

Data Set	Number of voxels	Number of faces	Time to reconstruct	Average Percentage Error (Per Pixel)
Dinosaur	21,000	24,000	510 seconds	10.1%
Dinosaur	52,500	147,000	6700 seconds	
Synthetic tori	700	1,500	260 seconds	11.9%
Synthetic tori	3,500	57,000	6,000 seconds	

4.1 Synthetic Imagery

The algorithm was tested on a synthetic scene of two texture-mapped, interlocking tori. The images were created by doing an off screen rendering of an Open Inventor scene graph consisting of two NURBS-based tori, each with a different texture with some high frequency elements. The tori were aligned such that the blue torus was aligned with the x-z plane and the yellow torus was aligned with the x-y plane. The cameras were all located in the same x-y plane above the tori facing down at the origin, which was located between the two tori. The reconstruction was computed with ten input images looking down at the scene. Figure 9 shows a comparison between the original input images and renderings of the reconstructed geometry from the same viewpoints.

Original Images	Reconstruction	Original Images	Reconstruction
			
			
			
			
			

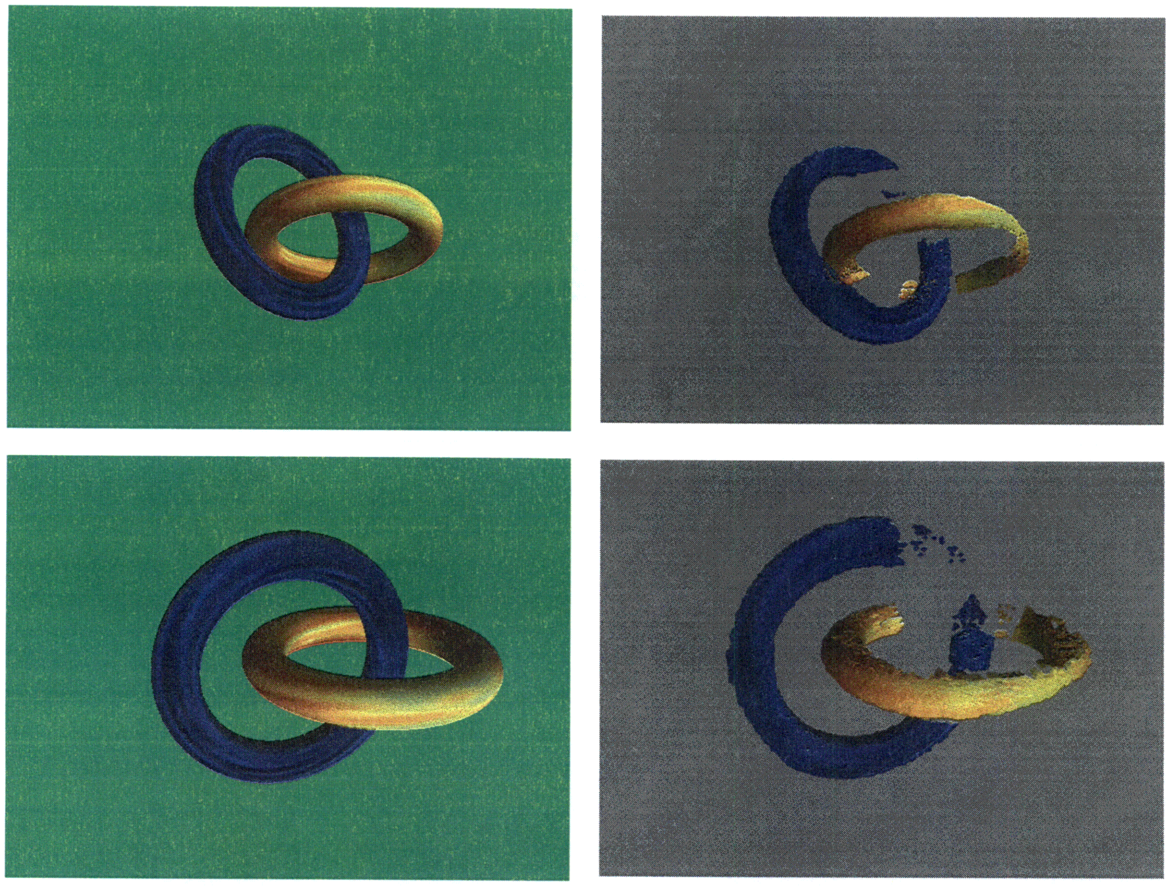
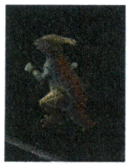
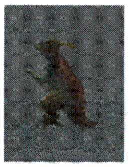
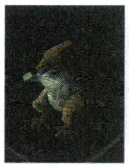
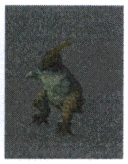
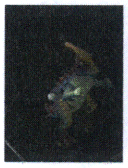
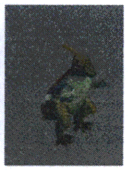
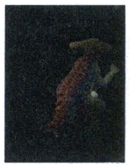
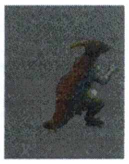
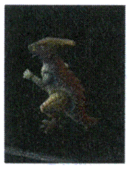
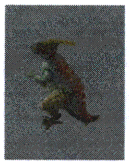
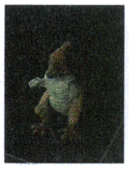
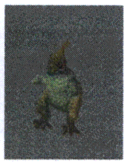
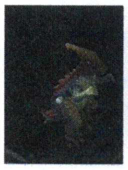
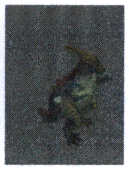
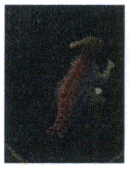
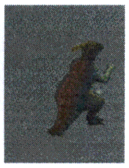
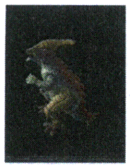
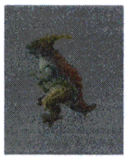
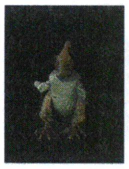
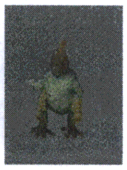
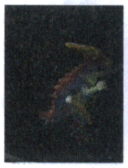
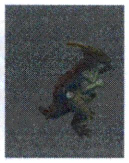
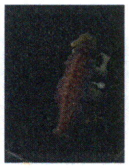
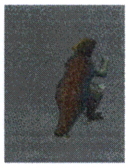
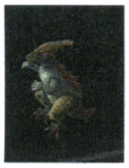
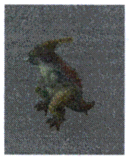
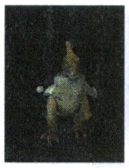
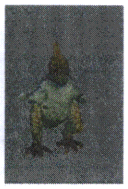
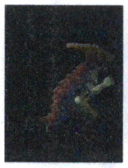
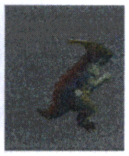
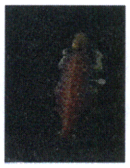
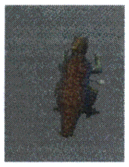
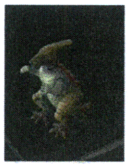
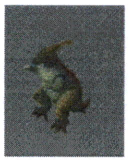
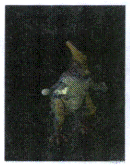
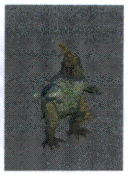
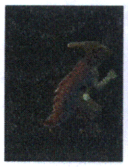
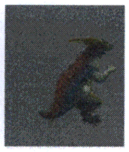
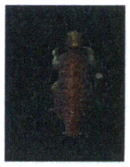
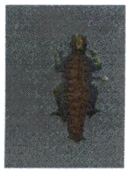


Figure 9: Renderings of original and reconstructed tori

The input images were given a green background to allow simple background segmentation. The reconstructions were computed on an SGI O2 workstation containing a 200 MHz MIPS R5000 processor with 256 MB of memory.

4.2 Real World Imagery

The algorithm was also run on a real world scene. The scene used was the same toy dinosaur used in [14], which was acquired by rotating the dinosaur using a computer-controlled turntable with a fixed camera and light source. Note that moving the turntable while keeping the light fixed creates the undesirable impression in the original images that the light is moving, which causes some surfaces to appear different from different viewpoints. Rotating the camera results in 21 images taken from cameras facing down at the dinosaur.

Original	**Reconst.**	**Original**	**Reconst.**	**Original**	**Reconst.**	**Original**	**Reconst.**
							
							
							
							
							

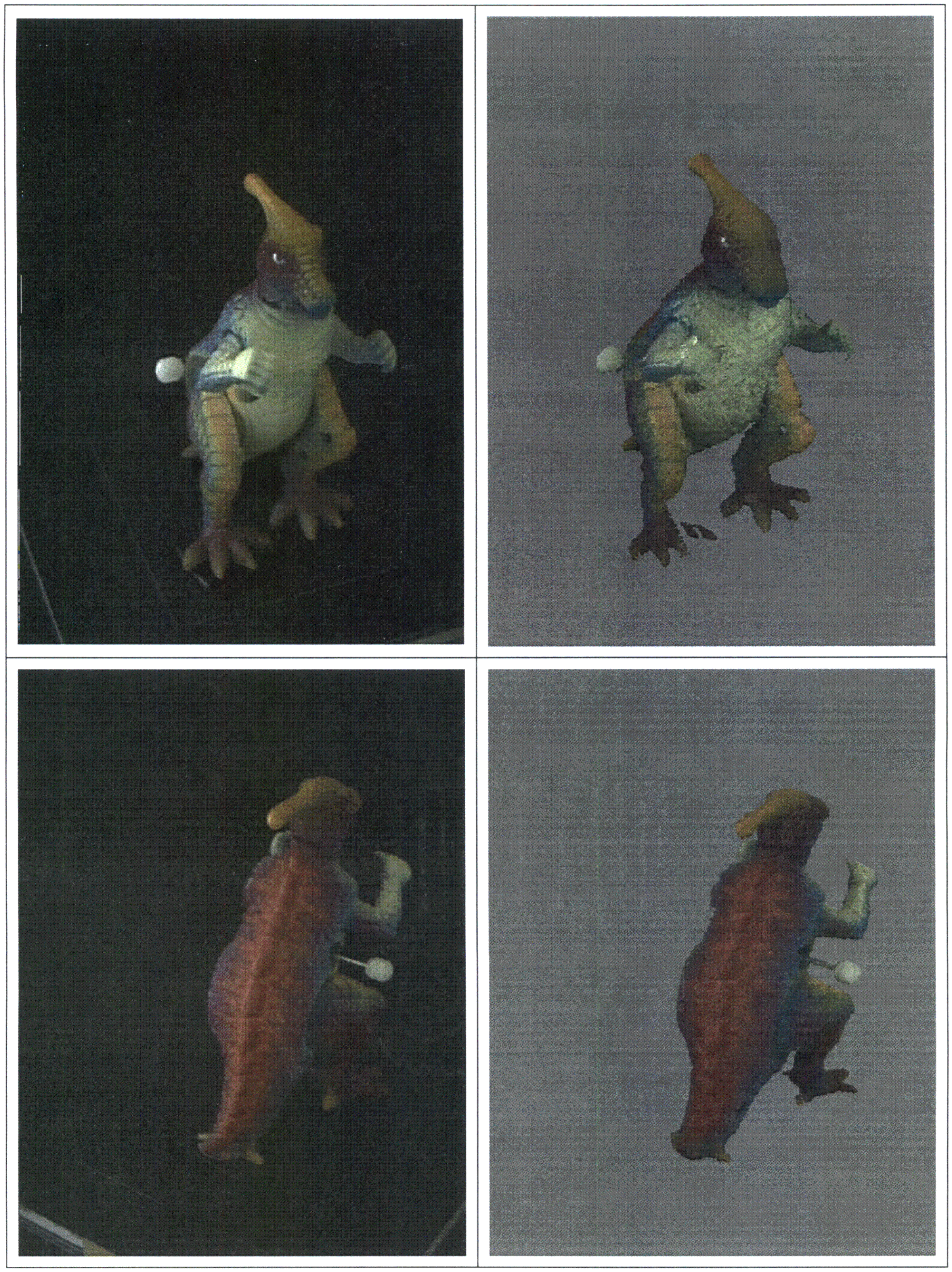


Figure 10: Dinosaur pictures and renderings of the reconstruction

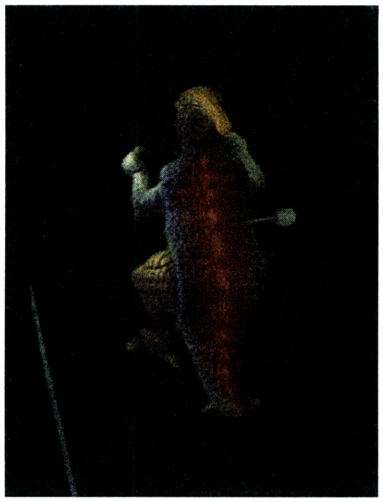
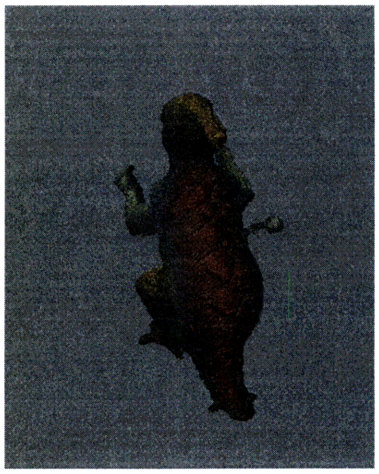
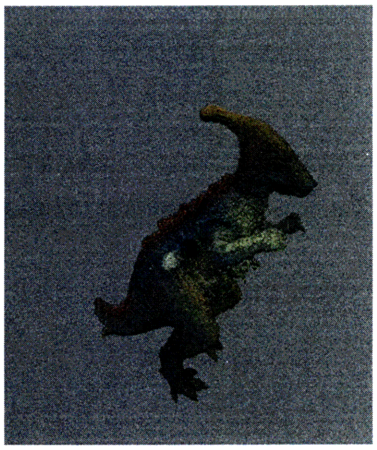
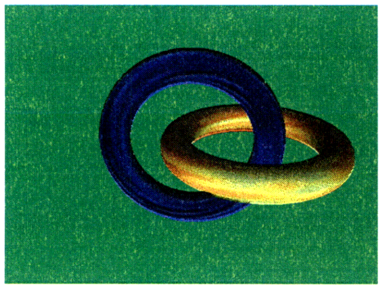
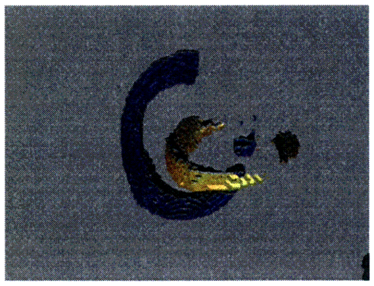
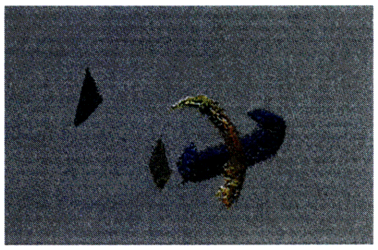
Unused input image	**Reconstruction at unused viewpoint**	**Synthetic views (not close to any input camera view)**
		
		

Figure 11: View of the reconstructions from synthetic viewpoints

The reconstructions were run at varying resolutions on a Hewlett-Packard HP9000-J282 workstation with dual 180 MHz processors and 512MB of memory using 20 of the 21 images. Figure 10 shows renderings of the reconstructions in comparison with the corresponding original images. As was the case in [14], fine details such as the wind-up knob and shaft and the pupils in the eye have been reconstructed.

5 Discussion

Blinking Cubes introduces a polygon-based approach for approximate reconstructions of scene structure from calibrated imagery. The use of a world-centric, surface-based representation accurately represents surfaces and structure from real and synthetic worlds. While inaccuracies do result, the model is polygon-based, and thus the manual process of refinement can be accomplished with common polygon-based modeling tools.

Segmenting the background from the foreground in the input images is an important issue both in terms of speed and accuracy. Both the real world dinosaur data set and the synthetic tori imagery have flat backgrounds that were avoided with a simple heuristic that eliminated geometry projecting to background pixels, which greatly decreased the time required to finish a reconstruction and avoided spurious reconstructions away from the object(s) of interest. However, this resulted in the loss of reconstruction detail around the edges of the scene object(s). Another issue of note occurs when part of a surface contains color variation that exceeds the resolution of the reconstruction. In this case, each image individually fails to correlate when in fact scene structure actually exists at the location being examined. To account for this, an additional correlation is attempted if the initial correlation fails. This high-frequency correlation compares the means and standard deviations of each image to see if the variation in each image has a similar statistical footprint.

Reconstructions are very sensitive to accurate occlusion. An initial test case of an extruded 'L' shape yielded very inaccurate results due to missed correlations at the corners. These missed occlusions allowed pixels to project through and prevent many corners. These missed occlusions allowed pixels to project through and prevent many subsequent reconstructions.

Both data sets demonstrated that determining the parameters for a quality reconstruction requires an empirical approach. In both cases, both the error threshold and the reconstruction volume resolution had to be adjusted multiple times to achieve optimal results. Not surprisingly, the real world imagery required a higher error threshold to achieve suitable results.

The dinosaur data set had problems with the background segmentation heuristic. The background color was similar to the colors of certain portions of the toy, and as a result, it caused some correlations to be missed. These incorrectly unmasked pixels subsequently "drilled" through the rest of the dinosaur model, creating what appears as a tear through the belly. However, Blinking Cubes was able to correctly reconstruct details such as the fingers and the shaft of the wind-up knob correctly.

The tori reconstruction had noticeable amounts of missed reconstruction. The majority of failures occurred in areas where pixels in one of the input images that projected to both tori, and the errors resulted in those regions for the following set of reasons: The reconstruction used a heuristic that prevented correlation if any of the hypothesized faces projected to a background colored pixel. Because the rendered images were not antialiased, this heuristic prevented surface reconstructions near the edges of the tori that otherwise could have passed the correlation test. In addition, both tori contained certain high frequency portions that were beyond the resolution capability of the reconstruction faces. These failed correlations left some pixels incorrectly unmasked, allowing them to project past their true surface to other surfaces and prevent correlations that would have succeeded with proper occlusion masking. Experiments with varying error thresholds and resolutions confirmed this reasoning.

6 Future Work

One potential way to increase accuracy is to use some type of view-dependent texture mapping, similar to [4], to make synthetic views more believable. Blinking Cubes also has the ability to identify areas where more input is desired. By including triangles that are not visible by input cameras and coloring them a default color, these faces can provide hints to a user about viewpoints where additional input would be most useful. The missed parts of the tori reconstruction highlight the sensitivity of the algorithm to proper occlusion masking and lend credibility to the application of a per-slice global optimization approach such as simulated annealing or tabu search. Because Blinking Cubes stores multiple values at each pixel, it can easily be modified to be part of a multi-pass algorithm. By storing these LDI values, subsequent iterations through the volume could refine the geometry by removing a voxel and examining the effects it has on the quality of the reconstruction. The layered depth pixels create a simple mechanism to determine what geometry is affected by the removal of other geometry.

Another improvement is to allow a local search for best-fit surfaces. All vertices are currently placed along a cube edge halfway between the two cube corners. Allowing these vertices to slide along their corresponding edges would increase the probability of generating surfaces with better correspondence. Blinking Cubes also generates a large number of triangles. The dinosaur data set generated approximately 150,000 triangles at its highest resolution, which is too complex to be viewed and manipulated in many graphics environments. The use of a mesh decimation algorithm can exploit the regularity of the reconstructed surfaces and greatly reduce the polygon count of reconstructions.

As discussed previously, the predetermined order in which to visit points in space is not well defined for many camera configurations. One approach to determining an order of iteration could be to use a greedy algorithm for reconstructing the point that has a clear line of sight to the most input cameras. As each point (or its associated geometry) succeeds or fails its correlation, the visibility of points along lines of sight between the current point and the cameras used in its correlation would be updated, and the most visible point would be the next examined for correlation. Once this method finishes, the model could then be refined and improved with some type of optimization approach such as the per-slice optimization approach discussed previously.

Blinking Cubes currently uses only triangles in its reconstructions, which is not representative of many surfaces found in the real world. The introduction of different, and especially more complex surface primitives or parametric surfaces could improve the accuracy with which surfaces are represented. Because Blinking Cubes uses oriented faces, more complex shading models could also be used to widen the variety of objects that can be reconstructed. In addition, using a more sophisticated approach to sampling images could help to improve reconstruction accuracy and reduce the effects of quantization errors. Finally, because Blinking Cubes, and in fact all correlation methods have difficulty with areas of high frequency, and in particular edges, an edge detection preprocess can help identify pixels corresponding to edges in which a correlation wouldn't normally be found but should occur.

While Blinking Cubes introduces a polygon-based approach for scene reconstruction, the ultimate goal for reconstruction methods is to develop a general-purpose framework for reconstructing scenes for use in a wide variety of applications. The work presented here can be viewed as a starting point for such a framework.

7 Acknowledgements

I extend a sincere thanks to both Tom Malzbender and Bruce Culbertson for their insights, critiques, criticisms, and encouragement throughout the course of this work. A special thanks to Tom for giving me the latitude to steer my research down a road less traveled. Thanks also goes to everyone in the Visual Computing Department at Hewlett-Packard Research Laboratories for their support, and to Hewlett-Packard for its consistent commitment to myself and to the VI-A internship program. A special thanks goes to Steve Seitz for the use of the dinosaur data set. I also thank Leonard McMillan for his valuable insight and editorial input.

8 References

1. Adelson E.H., Bergen J.R., "The Plenoptic Function and the Elements of Early Vision," Computational Models of Visual Processing, Chapter 1, Edited by Michael Landy and J. Anthony Movshon. The MIT Press, Cambridge, Mass. 1991.

2. Chen Shenchang Eric, Williams Lance, "View Interpolation for Image Synthesis", Proceedings of SIGGRAPH 93. In Computer Graphics Proceedings, Annual Conference Series, 1993, ACM SIGGRAPH, pp. 279-288.

3. Curless Brian, Levoy Marc, "A Volumetric Method for Building Complex Models from Range Images", Proceedings of SIGGRAPH 96. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 303-312.

4. Debevec Paul E., Taylor Camillo J., Malik Jitendra, "Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach", Proceedings of SIGGRAPH 96. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 11-20.

5. Gortler Steven J., Grzeszczuk Radek, Szeliski Richard, Cohen Michael F., "The Lumigraph", Proceedings of SIGGRAPH 96. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 43-54.

6. Gortler Steven J., He Li-wei, Cohen Michael F., "Rendering Layered Depth Images", Technical Report MSTR-TR-97-09, Microsoft Research, Advanced Technology Division, March, 1997.

7. Kanade T., Narayanan P.J., Rander P.W., "Virtualized Reality: Constructing Virtual Worlds from Real Scenes", Proceedings of the IEEE Workshop on Multimedia, volume 4, number 1, January - March 1997, pp. 34-47.

8. Kang S.B., Szeliski R., "3-D Scene Data Recovery using Omnidirectional Multibaseline Stereo" International Journal of Computer Vision, 1997.

9. Levoy Marc, Hanrahan Pat, "Light Field Rendering", Proceedings of SIGGRAPH 96. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 31-42.

10. Lorenson William E., Cline Harvey E., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm", Proceedings of SIGGRAPH 87. In Computer Graphics Proceedings, Annual Conference Series, 1987, ACM SIGGRAPH, pp. 163-169.

11. McMillan Leonard, Bishop Gary, "Plenoptic Modeling: An Image-Based Rendering System", Proceedings of SIGGRAPH 95. In Computer Graphics Proceedings, Annual Conference Series, 1995, ACM SIGGRAPH, pp. 39-46.

12. Sato Yoichi, Wheeler Mark D., Ikeuchi Katsushi, "Object Shape and Reflectance Modeling from Observation", Proceedings of SIGGRAPH 97. In Computer Graphics Proceedings, Annual Conference Series, 1997, ACM SIGGRAPH, pp. 379-387.

13. Seitz Steven M., Dyer Charles R., "Complete Scene Structure from Four Point Correspondences", Proceedings of the Fifth International Conference on Computer Vision, 1995, pp. 330-337.

14. Seitz Steven M., Dyer Charles R., "Photorealistic Scene Reconstruction by Voxel Coloring", Proc. Computer Vision and Pattern Recognition Conference, 1997, pp. 1067-1073.

15. Seitz Steven M., Dyer Charles R., "View Morphing", Proceedings of SIGGRAPH 96. In Computer Graphics Proceedings, Annual Conference Series, 1996, ACM SIGGRAPH, pp. 21-30.

16. Seitz Steven M., Kutulakos Kiriakos N., "Plenoptic Image Editing", Proceedings of the Fifth International Conference on Computer Vision, 1998, to appear.

17. Szeliski R., "Image Mosaicing for Tele-Reality Applications", Technical Report CRL 94/2, DEC and Cambridge Research Lab Technical Report, May 1994

54