To appear at VG '01, the International Workshop on Volume Graphics

June 21-22, 2001

A Survey of Methods for Volumetric Scene Reconstruction from Photographs

Greg Slabaugh[1]
Center for Signal and Image Processing
Georgia Institute of Technology

Bruce Culbertson[2]
Visual Computing Department
Hewlett-Packard Laboratories

Tom Malzbender[2]
Visual Computing Department
Hewlett-Packard Laboratories

Ron Schafer[1]
Center for Signal and Image Processing
Georgia Institute of Technology

ABSTRACT

Scene reconstruction, the task of generating a 3D model of a scene given multiple 2D photographs taken of the scene, is an old and difficult problem in computer vision. Since its introduction, scene reconstruction has found application in many fields, including robotics, virtual reality, and entertainment. Volumetric models are a natural choice for scene reconstruction. Three broad classes of volumetric reconstruction techniques have been developed based on geometric intersections, color consistency, and pair-wise matching. Some of these techniques have spawned a number of variations and undergone considerable refinement. This paper is a survey of techniques for volumetric scene reconstruction.

1. INTRODUCTION

We present a survey of techniques for volumetric scene reconstruction. In computer vision, scene reconstruction is an old and challenging problem whose aim is to create a 3D model of a scene, given 2D images of the scene. An important early application was robot navigation. Multimedia computing has generated renewed interest in the problem and has shifted the emphasis to generating new, "virtual" views of scenes. Applications include virtual reality, games, and special effects for motion pictures. In this paper, we focus on techniques developed primarily for reconstructing natural, real world scenes using off-the-shelf cameras. Therefore, we do not cover medical imaging or active light methods (such as those that use laser scanners) as these methods require specialized hardware.

Volumetric data representations have been gaining importance since their introduction in the early 70's in the context of 3D medical imaging [Greenleaf 70]. The exponential growth of computational storage and processing during the last three decades have enabled these representations to become practical alternatives to surface-based geometrical representations for many applications in computer graphics and scientific visualization [Kaufman 91]. Volumetric representations have also become an important tool in the field of computer vision. In particular, volumetric models provide a flexible and powerful representation for 3D objects inferred from (typically) multiple images of a scene.

Unfortunately there are some differences in the meaning of the word "volumetric" between the disciplines of computer graphics/scientific visualization and that of computer vision. In both cases, the term volumetric implies a representation that describes not only the surface of some region, but also the space that the region encloses. However in computer graphics, the term volumetric further implies a sampled representation. Various sampling patterns such as regular, non-isotropic, curvilinear, and unstructured are accommodated, and many of these allow extensions of signal and image processing methods to be applied. However, the term volumetric in the field of computer vision implies no such sampling; for example polyhedral representations are considered volumetric in this context. Papers such as [Pentland 90] and [Terzopoulos 91], described by vision researchers as involving volumetric representations [Fua 95], are examples of non-sampled representations. In this paper we restrict our usage of the term volumetric to imply sampled systems, those involving voxels, following the convention of the graphics and scientific visualization community.

Currently, several restrictions on the scenes and the input photographs are required to make reconstruction tractable. All the techniques described here require calibrated input images, which means we know where any 3D point in the scene projects in each image. ([Saito 99] and [Garcia 98] can exploit a somewhat weaker form of calibration.) Image calibration is itself a challenging problem with a large literature devoted to it [Hartley 00]. With the exception of [Szeliski 99] and [De Bonet 99], the techniques assume all surfaces are entirely opaque. The visual hull techniques require that foreground objects in the input images can be

[1] Atlanta, GA 30332. {slabaugh, rws}@ece.gatech.edu
[2] Palo Alto, CA 94306. {bruce_culbertson, tom_malzbender}@hp.com

segmented from the background; the rest of the techniques assume that surfaces are Lambertian (they reflect light equally in all directions) or are nearly so.

The rest of this paper is organized as follows. Section 2 discusses methods for volumetric reconstruction of visual hulls, which are based on geometric intersection. Section 3 considers voxel coloring methods, which rely upon the consistency of colors observed across images. Finally, Section 4 discusses a class of techniques based on pair-wise matching.

2. VOLUMETRIC VISUAL HULLS

The earliest attempts at volumetric model reconstruction from photographs are those that approximate the *visual hull* of the imaged objects. This technique is also referred to as *volume intersection* in the vision literature. The visual hull of an object can be described as the maximal shape that gives the same silhouette as the actual object for all views outside the convex hull of the object [Laurentini 94]. Volume intersection methods use a finite set of viewpoints, and compute what we will call the *inferred visual hull*, as shown in Figure 1. Typically one starts with a set of source images that are simply projections of the object onto *N* known image planes. Each of these *N* images must then be segmented into a binary image containing foreground regions to which the object projects; everything else is background. If these foreground regions are then back-projected into 3D space and intersected, the resultant volume is the inferred visual hull of the object.

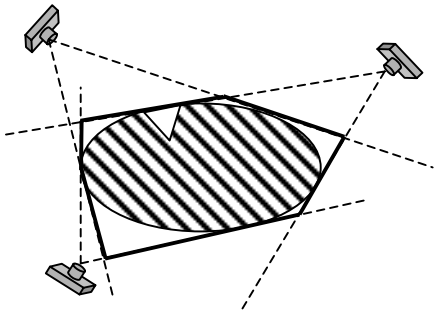


Figure 1. Object (hatched) and its inferred visual hull (bold).

The inferred visual hull has several interesting properties. First, although it is only an approximation to the true shape of the object, it is guaranteed to enclose the object. Second, in 3D the inferred visual hull of an object can be a better or worse approximation of the object than the convex hull depending on the geometry of the object and the range of the viewpoints. Third, the size of the inferred visual hull decreases monotonically with the number of images used. However, even when an infinite number of images are used, not all concavities can be modeled with a visual hull. For the rest of this paper we will use the term "visual hull" to mean the inferred visual hull computed from *N* images.

The earliest work reporting a volumetric representation of the visual hull is due to Martin and Aggarwal [Martin 83]. They recommend simple intensity thresholding of each input image to perform segmentation into object foreground and background. A connected component analysis of the resulting binary image yields the silhouette. An initial *parallelogram structure* is then extracted by combining orthographic back-projections from multiple images. This is further processed into a *volume segment* representation, which is a set of line segments parallel to one axis of their coordinate system. This representation is then further processed into a surface description when desirable.

Further work in Aggarwal's lab led to explorations of octree representations for the visual hull [Chien 84, Chien 86]. The method starts with three binary images from orthogonal viewing directions. These are converted to three quadtree representations, which are subsequently merged into an octree representation of the visual hull. A disadvantage of this approach is the limitation on the number of input images and the strict requirement on the orthogonality of their optical axes. This limits the fidelity of the reconstructed volume. Additionally, the method appears limited to cameras with parallel projection transforms. Contemporary work by Shneier et al. [Shneier 84] also proposes building octrees from segmented images, however no examples were given since their implementation was not complete. For the special case of source images being available from 13 prescribed orthographic viewpoints, an octree reconstruction is described in [Veenstra 86].

An early implementation on a PDP11 by [Massone 85] used actual photographic input from vidicon cameras to carve the visual hull from regularly sampled voxels. This method is flexible enough to handle both perspective and parallel projections.

[Potmesil 87] also reconstructs an octree representation of objects from multiple images, now handling arbitrary viewpoints and perspective projections. He divides the task into three components. First he generates conic octree volumes from silhouettes of the objects. Second, he combines sequences of these conic octree volumes into a global model. Third, a 3D connected components algorithm is used to label individual objects. Surface normals and textures are then mapped onto the object with surface normals computed from the local octree topology. Textures are sampled from the original source images, averaging being performed when multiple images are available for a particular surface point. [Srivastava 90] also constructs an octree from arbitrary perspective images. They also propose thresholding for segmentation and then approximate the boundary of the silhouette polygonally. The polygons are then decomposed into convex components and efficient octree intersection tests with the back-projections are employed. Source images presented are computer generated, not photographic.

[Szeliski 93] builds volumetric models directly from actual photographs. Although the approach is similar to the work of Potmesil, there are numerous significant differences. First, where Potmesil builds a separate octree per image and then combines them, Szeliski refines a single octree model with each successive frame. This allows significant increases in processing speed. Additionally Szeliski is the first to address many practical issues in this context, such as performing

adaptive background subtraction and morphological operations during the segmentation stage, and automatic determination of turntable orientation. His work is also leveraged in the Lumigraph system [Gortler 96] for finding approximate geometry to improve image quality during scene rendering. A variant of the octree representation for volume intersection is given by [Garcia 98]. He uses a projective octree representation, defined by selecting two images with optical axes approximately at right angles to each other. Projective projections of these reference images define a 3D coordinate system on which the octree is defined. Interestingly, since only the fundamental matrix [Luong 96] is computed between these images by selecting correspondences, the actual geometrical deformation of the octree is not explicit. Binary segmentations from additional images can be incorporated by computing the trilinear tensor [Shashua 95] between such an image and the two reference images. A similar projective grid space is also used for voxel coloring in [Saito 99] and is discussed later in the paper.

[Seitz 95] presents a novel Hough-like voting scheme that back-projects image features into a volumetric space. Although explicit voxels are not maintained, bins corresponding to regions of space containing features are allocated as needed. A contribution of the work is an implicit formulation that permits reconstruction of both point and line features within a common three-dimensional parameter space. The output of this method is a 3D representation of features, and an additional process would be needed for model reconstruction. This also applies to the work of [Collins 96] who back-projects features onto a plane that sweeps through space. Like Seitz's work, a full volume is not maintained at any point in time to save memory.

Work by Fromherz [Fromherz 94] sculpts a volume from a voxel array that is uniformly spaced with voxel projections on the scale of pixels in the source images. Results on a mannequin are verified by comparing with computed tomography scans. An automatic binary segmentation is used on the source images. After computing this volume representing the visual hull, a further refinement is performed that uses luminance information in the original source images [Fromherz 95]. At each iteration, surface voxels are projected into sequential image pairs from a rotation sequence. If the luminance of those pixels differ an amount greater than a threshold, the voxel is removed from the model. This work is the first example of luminance-based carving that we are aware of.

During the later 90's volumetric visual hulls were computed from video streams originating from multiple cameras for the first time. [Moezzi 96] describes a system of 17 cameras centered on a 1m x 1m x 2m dynamic scene. Each frame is segmented into a binary image employing background subtraction techniques with careful control of lighting. Volume intersection is employed offline to construct a visual hull model composed of voxels that measure 1 cubic cm. Subsequent isosurface extraction yields a polygonal surface where colors are assigned to each polygon from area-weighted contributions from the source images. Improved surface coloring methods are introduced in [Moezzi 97].

3. VOXEL COLORING METHODS

3.1. Color Consistency

Many reconstruction algorithms use *color consistency*, introduced by [Seitz 97], to distinguish surface points from other points in a scene. As shown in Figure 2, cameras with an unoccluded view of a non-surface point see surfaces beyond the point, and hence inconsistent (i.e., dissimilar) colors, in the direction of the point. The consistency of a set of colors can be defined as their standard deviation or, alternatively, the maximum of the L_1, L_2, or L_∞ norm between all pairs of the colors. Any of these measures can be computed for the colors of the set of pixels that can see a voxel; the voxel is considered to be on a surface if the measure is less than some threshold.

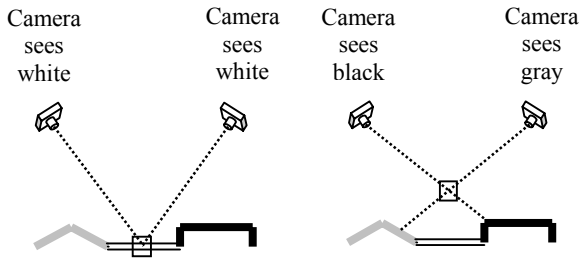


Figure 2. Color consistency can be used to distinguish points on a surface from points not on a surface. On the left, two cameras see consistent colors at a point on a surface. On the right, the cameras see inconsistent colors at a point not on the surface.

Real world scenes often include surfaces with abrupt color boundaries. Voxels that span such boundaries are likely to be visible from a set of pixels that are inconsistent in color. Hence, for such voxels, color consistency can fail as a surface test. This problem can be solved with an adaptive threshold that increases when voxels appear inconsistent from single images.

3.2. Restricted Camera Placement

[Fromherz 95] performed reconstructions by combining a consistency-based surface test, except using only luminance, with volume intersection. Seitz and Dyer [Seitz 97] demonstrated that a sufficiently colorful scene could be reconstructed using full-color-based consistency alone, without volume intersection. They called their algorithm *Voxel Coloring*.

The Voxel Coloring algorithm begins with a reconstruction volume of initially opaque voxels that encompasses the scene to be reconstructed. As the algorithm runs, opaque voxels are tested for color consistency and those that are found to be inconsistent are *carved*, i.e. made transparent. The algorithm stops when all the remaining opaque voxels are color-consistent. When these final voxels are assigned the colors they project to in the input images, they form a model that closely resembles the scene.

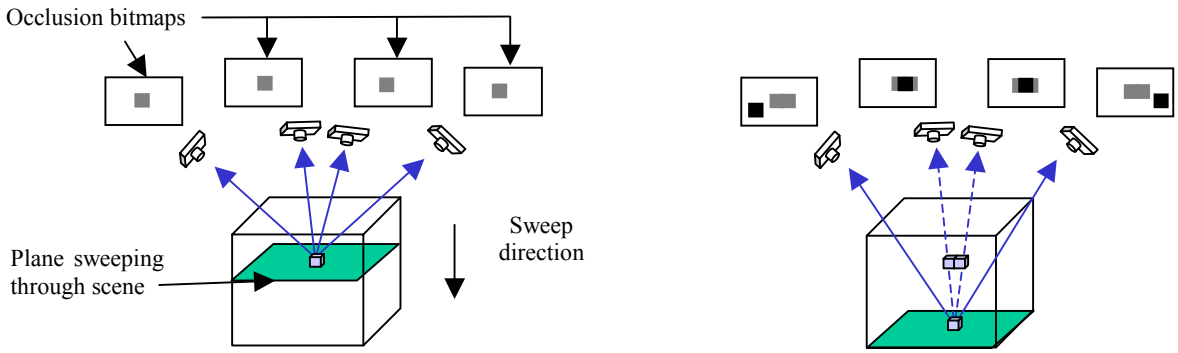


Figure 3. Using occlusion bitmaps. On the left, a voxel is found to be consistent, and a bit in the occlusion bitmap is set for each pixel in the projection of a consistent voxel into each image. On the right, visibility of the lowest voxel is established by examining the pixels to which the voxel projects. These pixels are shown in black. If the occlusion bits have been set for these pixels, then the voxel is occluded, as is the case for the two middle cameras.

As Voxel Coloring progresses, opaque voxels occlude each other from the input images in a complex and constantly changing pattern. To test the color consistency of a voxel, its visibility (the set of input image pixels that can see it) must first be determined. Since this is done many times during a reconstruction, it must be performed efficiently. Calculating visibility is a subtle part of algorithms based on color consistency and several interesting variations have been developed.

To simplify the computation of voxel visibility and to allow a scene to be reconstructed in a single scan of the voxels, Seitz and Dyer imposed what they called the *ordinal visibility constraint* on the camera locations. It requires that the cameras be placed such that all the voxels are visited in a single scan in near-to-far order relative to every camera. Typically, this condition is met by placing all the cameras on one side of the scene and scanning voxels in planes that are successively further from the cameras. Thus, the transparency of all voxels that might occlude a given voxel is determined before the given voxel is checked for color consistency. This insures that the visibility of a voxel stops changing before it needs to be computed, which is important since every voxel is visited only once. An occlusion bit map, with one bit per input camera pixel, is used to account for occlusion. These bits are initially clear. When a voxel is found to be consistent, meaning it will remain opaque, all the occlusion bits in the voxel's projection are set, as shown in Figure 3. The visibility set of a voxel is simply the pixels in the voxel's projection whose occlusion bits are clear.

The runtime for Voxel Coloring is related to the number of voxels. [Prock 98] achieves as much as a 40× speedup of the algorithm using multiple voxel resolutions. First, a rapid reconstruction is performed using coarse voxels. Some voxels will be mostly, but not entirely, unoccupied by scene objects. These voxels are likely to be carved yet, when subdivided into smaller voxels, they may contain small voxels that are mostly occupied. Hence, carved voxels that are adjacent to uncarved ones are added back into the model, i.e. are made opaque. Then the model is subdivided, usually by replacing each opaque voxel with eight smaller ones. This is used as the starting point for another pass of Voxel Coloring. These steps are repeated as long as warranted by the image resolutions.

3.3. Arbitrary Camera Placement

Voxel Coloring is elegant and efficient. However, the ordinal visibility constraint is a significant limitation. Since the voxels can be ordered from near to far relative to all the cameras, the cameras cannot surround the scene. So, some surfaces will not be visible in any image and hence cannot be reconstructed. Because it is often desirable to obtain a model that resembles the scene from every direction, several variations of Voxel Coloring have been developed to circumvent this limitation. If we surround the scene with cameras, we give up the ordinal visibility constraint. Without the constraint, there is no order in which to scan voxels that guarantees their visibility will not change after we check their color consistency. Hence, algorithms that allow arbitrary camera placement must test voxels repeatedly for consistency until their visibility stabilizes.

Figure 4 gives the general approach for Voxel Coloring algorithms that allow arbitrary camera placement. In the inner loop, the visibility of voxels is found, their consistency is checked, and they are carved if they are found to be inconsistent. If one voxel is carved, the visibility of other voxels potentially changes, invalidating any consistency tests they may have passed. Hence, there is an outer loop that repeats the consistency checking until no carving occurs in the inner loop. No carving occurs on the final iteration of the outer loop so no testing is invalidated and the final set of opaque voxels is guaranteed to be consistent.

When the algorithm in Figure 4 begins to run, the model bears little resemblance to the scene. Yet, the algorithm computes the visibility for voxels, and carves those found to be inconsistent, based on this model. It is reasonable to wonder if the algorithm might fail due to carving voxels early on that would be color consistent in the final model. [Kutulakos 98] has shown that, in fact, this cannot happen if a suitable consistency measure is used. The measure must be *monotonic*: if it finds a set of pixels to be inconsistent, then it will find any superset of those pixels to also be inconsistent. Since the algorithm only changes opaque voxels to

```
1  set all voxels opaque
2  loop {
3         AllVoxelsConsistent = TRUE
4         for every opaque voxel V {
5                find the set S of input image pixels from which V is visible
6                if S has consistent color {
7                       assign V the average color of all pixels in S
8                } else {
9                       AllVoxelsConsistent = FALSE
10                      set V to be transparent
11               }
12        }
13        if AllVoxelsConsistent = TRUE
14               quit
15 }
```

Figure 4. Pseudocode for Voxel Coloring algorithms with unconstrained cameras.

transparent and never vice versa, remaining opaque voxels can only become more visible as the algorithm runs and the pixels that can see a voxel at one point in time will be a subset of those that see the voxel at any later time. Thus, if the monotonic consistency measure ever finds a voxel to be inconsistent, the voxel will also be inconsistent in the final model. Therefore, the algorithm never carves a voxel it shouldn't—one that would be consistent in the final model—and so we say carving is *conservative*. Furthermore, Kutulakos and Seitz proved that the algorithm finds the unique color consistent model that is a superset of any other consistent model. They call this unique model the *photo hull*.

3.3.1. Space carving

[Kutulakos 98] describes an implementation of Figure 4 called *Space Carving*. It always scans voxels for color consistency by evaluating a plane of voxels at a time, as is often done with Voxel Coloring. Unlike Voxel Coloring, Space Carving uses multiple scans, typically along the positive and negative directions of each of the three axes. Space Carving forces the scans to be near-to-far, relative to the cameras, by using only images whose cameras have already been passed by the moving plane. Thus, when a voxel is evaluated, the transparency is already known of other voxels that might occlude it from the cameras currently being used. Because carving is conservative, the set of uncarved voxels is a shrinking superset of the desired color-consistent model as the algorithm runs.

Space Carving achieves the goal of allowing arbitrary camera placement but only implements the pseudocode in Figure 4 approximately. Space Carving never carves voxels it shouldn't but it is likely to produce a model that includes some color-*in*consistent voxels. This is because, during scanning, cameras that are ahead of the moving plane are not used for consistency checking, even when the voxels being checked are visible from those cameras. Hence, the color consistency of a voxel is, in general, never checked over the entire set of images from which it is visible. (A later paper, [Kutulakos 00b], describes additional bookkeeping that enables Space Carving to compute visibility exactly.)

In a practical setting, the camera calibration is not precisely known, so the visible pixels to which a voxel projects in an image can contain incorrect pixels. [Kutulakos 00a] presents a variation of space carving called "approximate space carving" that addresses this problem of inaccurate camera calibration. When evaluating a voxel's consistency, this method considers a disk of radius r in each image, centered at the projection of the voxel center. If there is a pixel color that appears in all disks, then the voxel is said to be r-consistent, and remains in the volume. Otherwise, the voxel is carved. Using a larger value of r allows one to reconstruct the scene with poorly calibrated cameras. By reconstructing the scene with a range of decreasing values of r, a set of nested reconstructions are found, each of which is a tighter bound on the true 3D geometry being reconstructed.

3.3.2. Generalized Voxel Coloring

[Culbertson 99] describes a reasonably efficient and simple implementation of Figure 4, called Generalized Voxel Coloring (GVC), that computes visibility exactly and hence, yields a color consistent model. They provide experimental results that show that exact visibility, when compared with the approximate visibility computed by Space Carving, can result in better looking reconstructions that are numerically more consistent with the input images.

Two variants of the algorithm, called GVC-IB and GVC-LDI, have been developed. They use different data structures, called item buffers (IBs) [Weghorst 84] and layered depth images (LDIs) [Max 96] [Shade 98], to compute the visibility of voxels. See Figure 5. An item buffer records, for every pixel in an image, the surface voxel that is visible from the pixel. An LDI records, for every pixel in an image, a depth-sorted list of all surface voxels that project to the pixel. The information in an LDI is a superset of the information in an item buffer and generally consumes considerably more memory. As GVC-IB runs, an item buffer for each input image can be computed by rendering, using Z-buffering, the current set of surface voxels to the image viewpoint. Unique voxel identifiers are rendered to the item buffer in place of the colors normally rendered with Z-buffering. The rendering can be performed using software or a hardware graphics accelerator. LDIs are computed similarly but, instead of using Z-buffering to find the voxel closest to each pixel, all surface voxels that project to a pixel are inserted in sorted order in a list for the pixel.

The visible pixels to which a voxel projects in an image are found as follows. First, the voxel is scan-converted to find the

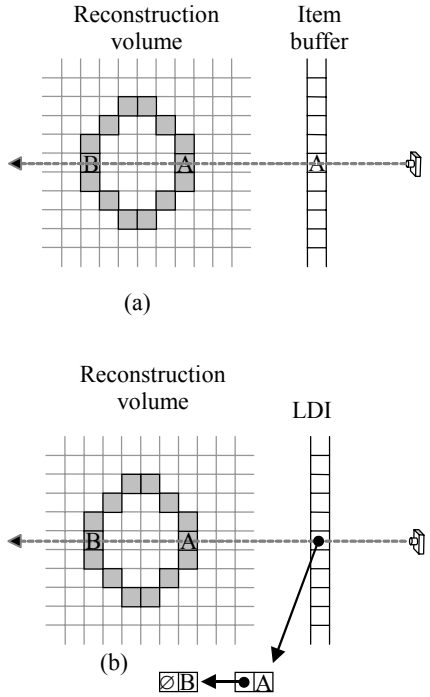


Figure 5. The two variants of GVC use different data structures, called *item buffers* and *layered depth images* (LDIs), to compute the visibility of voxels. In (a), an item buffer records, for every pixel in an image, the surface voxel that is visible from the pixel. In (b), an LDI records, for every pixel in an image, a depth-sorted list of all surface voxels that project to the pixel.

pixels in the voxel's projection. In the GVC-IB case, the voxel is visible from the subset of these pixels whose item buffer values match the voxel's identifier. The same method can be used to find the visibility of a voxel in GVC-LDI since the nearest voxel in a pixel's LDI list is the same voxel that would be recorded for the pixel in an item buffer.

Since carving a voxel can change the visibility of other voxels, carving invalidates the item buffers. Hence, it may seem reasonable to compute new item buffers whenever a voxel is carved. This indeed produces correct results but is very slow. Fortunately, because carving is conservative, the item buffers can be updated less frequently, and the resulting out-of-date item buffers can be used for carving, without carving voxels that should not be carved. On the last iteration, the item buffers remain valid so the final set of opaque voxels is consistent. It is convenient and efficient to update the item buffers in the outer loop, after line 3 in Figure 4.

Carving also invalidates an LDI. However, an LDI can be updated incrementally with minimal computation. Voxels may be added to, or deleted from, an LDI by first finding the pixels in the voxel's projection and then adding or deleting the voxel from the LDI lists for those pixels. Hence, LDIs are updated immediately after carving occurs in GVC-LDI. The chief benefit of using LDIs, which compensates for their memory needs, is that they make it possible to tell precisely which voxels change visibility after a voxel is carved. When LDIs are updated, any voxel that moves into or out of the "nearest" position in a pixel's LDI list has different visibility after the update. It is only necessary to recheck a voxel's consistency if its visibility changes. Using item buffers, there is no efficient way to determine which voxels have increased visibility after carving occurs, so all voxels in the current model must have their consistency rechecked. Thus, GVC-LDI performs many fewer consistency checks during a reconstruction than GVC-IB, but this comes at the expense of increased memory use. A GVC-IB reconstruction is shown in Figure 9.

3.3.3. Multi-Hypothesis Voxel Coloring

[Eisert 99] has proposed a multi-hypothesis voxel coloring technique. A *hypothesis* is a possible coloring of a voxel. Their approach begins with a *hypothesis assignment* step, which identifies a set of hypotheses for each voxel. Then their algorithm narrows down the hypotheses during a *hypothesis removal* step, which carves inconsistent voxels. The surface voxels that remain constitute the volumetric reconstruction.

Hypothesis assignment begins by determining the color of the pixel to which a voxel center projects into each image. These pixel colors are compared for all pairs of views. If at least two cameras see a consistent color for the voxel, a hypothesis is assigned to the voxel. Consistency is determined by thresholding a distance measure in RGB space [Eisert 99] or normalized RGB space [Steinbach 00a] of the pixels. This process is executed for each voxel. During hypothesis assignment, there is no reconstructed geometry yet, so no occlusion information is available, as shown in Figure 6. Consequently, the hypothesis assignment step may assign hypotheses that do not correspond to the correct color of the surface being reconstructed.

Hypothesis removal takes occlusion into account to remove such hypotheses. For a given view, the voxel space is traversed in an occlusion-compatible direction [Steinbach 00b]. A visible voxel is projected into the image, and the pixel to which the voxel center [Eisert 99] (or pixels in the footprint around the voxel center [Steinbach 00b]) projects is compared with the voxel's hypotheses. The hypotheses that are not consistent are removed, and this process is repeated for the other viewpoints. If all of a voxel's hypotheses are removed, then no consistent color is observed across the images that have visibility of the voxel, and the voxel is carved. Carving a voxel changes visibility of other voxels that are then processed. The algorithm iterates over the surface voxels and all images until no more hypotheses can be removed, resulting in a photo hull.

Multi-hypothesis voxel coloring is quite similar to Voxel Coloring, Space Carving, and GVC. The key difference is that the decision to carve a voxel in these methods is made using all images simultaneously. In contrast, multi-hypothesis voxel coloring algorithms have an advantage in that hypothesis removal (and ultimately carving) is performed one image at a time. This simplifies visibility determination, since during hypothesis removal, the voxel space can be scanned

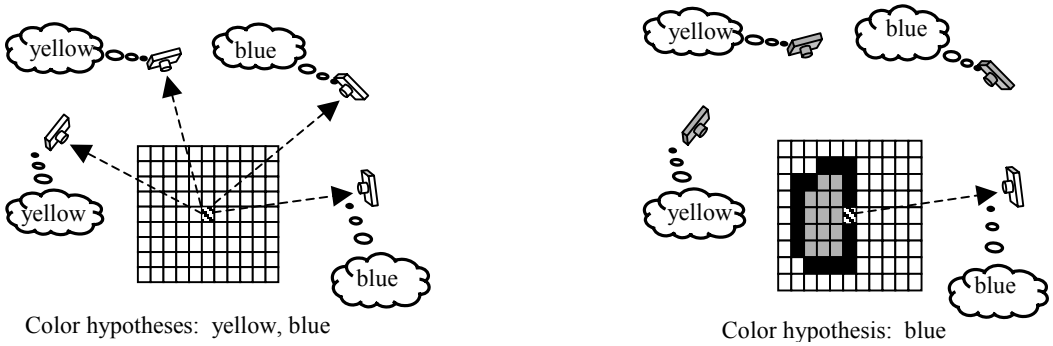


Figure 6. Multi-hypothesis voxel coloring. Hypothesis assignment for a voxel is shown on the left, where the striped voxel is projected into each image, and hypotheses are stored for the voxel. On the right, hypotheses that are inconsistent with the active camera's observation are removed. (The active camera is given a white color). Hypothesis removal considers the visibility of the scene. Thus, for the configuration of the voxel space on the right, the cameras that observe yellow for the voxel will not be considered.

front-to-back for *one* camera at a time. Consequently, occlusion bitmaps, the very simple and memory efficient data structures used in Voxel Coloring, can always be used to establish the exact visibility of the scene for arbitrary camera placement. However, the price to be paid for this convenience is some extra computation, as hypotheses are assigned to all voxels in the voxel space, including *interior voxels*, those that are inside surfaces. In GVC, for example, interior voxels never become visible and are therefore not processed.

3.4. Volumetric Optimization

3.4.1. Opaque voxels

The voxel coloring methods described above determine the consistency of a voxel by thresholding a color matching metric. This approach is intuitive and easy to implement. When reconstructing scenes with near Lambertian surfaces without abrupt color boundaries, and using accurately calibrated cameras, one can use a small threshold. This will produce a reconstruction that closely matches the true 3D geometry of scene surfaces. However, when these ideal conditions are not met, it is necessary to increase threshold so that scene surfaces reconstruct properly.

In general, there is not a single threshold that is ideal for reconstructing all surfaces in the scene. For a given threshold, some surfaces could likely be more accurately reconstructed with a lower threshold. But lowering the global threshold can cause other surfaces to become carved that should be in the final reconstruction. Consequently, the reconstructed model computed by voxel coloring algorithms tends to be larger (fatter) than necessary.

[Slabaugh 00b] presents a volumetric optimization method that refines a reconstruction to minimize *reprojection error*, the difference between a synthetic view formed by reprojecting the reconstruction to a camera and its reference image, summed over all views. The refinement effectively produces a spatially varying consistency threshold, tuned for each voxel in the scene, and results in a reconstruction that is typically a tighter fit to the true scene geometry. The volumetric optimization attempts to remove voxels, as well as add them, if it yields a more favorable surface reconstruction. The authors explore greedy and simulated annealing methods to perform the optimization.

3.4.2. Non-opaque voxels

[Szeliski 98] describes a reconstruction algorithm that uses partial opacity to address the problem of *mixed pixels*, pixels that fall on an occlusion boundary and have a mixture of foreground and background colors. A virtual camera viewpoint is chosen and used to define a discretized 3D disparity space whose coordinates are the two virtual image coordinates plus disparity relative to the virtual image. For each sample point in this space, the set of input image pixels that project to the point is found. At this stage, occlusion is disregarded. These sets of pixels are checked for color consistency and the set of sample points where there is high consistency is used as an initial surface. This initialization process and the use of a disparity space force the input cameras to be on one side of the scene. The initial surface is considered to be opaque. Next, occlusion is taken into account and the consistency of sample points is checked repeatedly, as is done in some of the variations of Voxel Coloring, until a color and binary opacity are assigned to every sample point. Finally, the colors and opacities are refined through optimization. The optimization favors smooth surfaces and encourages, but does not require, binary opacities. Furthermore, the optimization favors colors and opacities that match the input images when composited and projected to the camera viewpoints.

The Roxels algorithm [De Bonet 99] also attempts to determine continuous opacity values. It is more general than [Szeliski 98] in that it allows arbitrary camera placement and reconstructs objects that are semi-transparent. Roxels assigns colors and opacities to a uniform voxel space. The voxels can be rendered to the input image viewpoints using compositing with the over operator. Roxels also attempts to minimize the reprojection error. De Bonet and Viola point out that direct optimization of the colors *and* opacities to minimize the reprojection error is impractical because of the number of parameters and the fact that the error is a nonlinear function of the opacities. They observe, however, that the image pixels

are, in fact, linear combinations of the colors of the voxels along their rays. They call the coefficients of the linear combinations *responsibilities*, which can be found relatively efficiently. De Bonet and Viola use an iterative algorithm that solves for the responsibilities and then uses the responsibilities to estimate the opacities.

3.5. Alternate Voxel Spaces

3.5.1. Projective Grid Space

Camera calibration can be tedious and time-consuming. To reduce the amount of effort it takes to calibrate the cameras, [Saito 99] proposes voxel coloring in *projective grid space*. This is a voxel space where the voxels have a non-uniform shape based on the epipolar geometry relating two views. In their method, voxels increase in size in proportion to their distance from the basis views. Once the projective grid space is specified, it is easy to project points in projective grid space into any of the remaining viewpoints. [Kimura 99] later develops a similar approach that uses three basis images.

3.5.2. Volumetric Warping

The voxel-based reconstruction methods discussed in this paper are effective at reconstructing objects that are relatively close to the cameras. Applying them to large-scale scenes that contain surfaces very far from the cameras can become challenging, as doing so may require an unwieldy number of voxels that becomes prohibitive to process. Furthermore, it may be preferable to model far away objects with lower resolution voxels. Thus, one might like a spatially adaptive voxel size that increases away from the cameras.

[Slabaugh 00a] presents a method that warps the voxel space so that such scenes can be modeled without an excessive number of voxels. The method divides the voxel space into two regions; an interior space and an exterior space, as shown in Figure 7. The volumetric warp does not affect the voxels in the interior space, providing backward compatibility with previous voxel coloring algorithms, and allowing reconstruction of objects in the foreground at a fixed voxel resolution. Voxels in the exterior space are warped according to a warping function that changes the size of the voxel based on its distance from the interior space. The further a voxel in the exterior space is located from the interior space, the larger its size, as shown in Figure 7. Voxels on the outer shell of the exterior space have coordinates warped to infinity, and have infinite volume. Note that while the voxels in the warped space have a variable size, the voxel space still has a regular 3D lattice topology. They then reconstruct a large outdoor scene with GVC algorithm using this warped voxel space, shown in Figure 10.

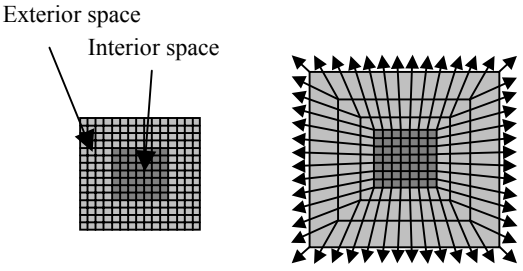


Figure 7. Volumetric Warping. Pre-warped (left) and warped (right) voxel spaces are shown in two dimensions. The voxel space is divided into two regions; an interior space shown with dark gray voxels, and an exterior space shown with light gray voxels. Before the warping is applied, both regions consist of voxels of uniform size. The warping does not affect the voxels in the interior space, while the voxels in the exterior space increase in size further from the interior space. The outer shell of voxels in the warped voxel space gets warped to infinity. These voxels are represented with arrows in the figure.

3.5.3. Two Linked Voxel Spaces

Any of the techniques discussed in this paper can reconstruct a time-varying scene recorded by multiple cameras by executing the algorithm once for each instant of time. However, such an approach does not take advantage of temporal coherency. [Vedula 00] presents a voxel coloring method that links two time-consecutive 3D voxel spaces together, forming a 6D space. A point in this space, a 6-dimensional element called a *hexel*, is a voxel in the 3D voxel space at time t_0 linked to another voxel in the 3D voxel space at time t_1. The goal of their method is to simultaneously reconstruct the shape and motion of the scene for the two instants of time.

To do so, they extend the notion of photo-consistency to this six-dimensional domain. A hexel is said to be photo-consistent if the two voxels that constitute the hexel project to pixels of approximately the same color in all of the images in which they are visible. This definition of photo-consistency is stronger than that used in voxel coloring, for it requires the voxel project to similar colors (1) from all viewpoints, as well as (2) at both time instants. The authors then present an approach to sweep through the 6D space to carve hexels. They show that this stronger consistency measure can carve voxels that do not have consistent motion, producing better results than are possible if a separate reconstruction is performed at each time instant. The output of their method is two reconstructed voxel spaces, one at time t_0 and another at time t_1, as well as the motion of surfaces (scene flow) between the two voxel spaces.

4. VOLUMETRIC PAIR-WISE FEATURE MATCHING

4.1 Image Space Methods

A common approach to 3D scene reconstruction relies upon the challenging task of robustly matching features between image pairs. These methods typically employ normalized cross-correlation along epipolar lines. Several authors have pursued volumetric representations to assist in this task, typically with image coordinates for two axes and a disparity hypothesis being the third axis. The earliest example is given

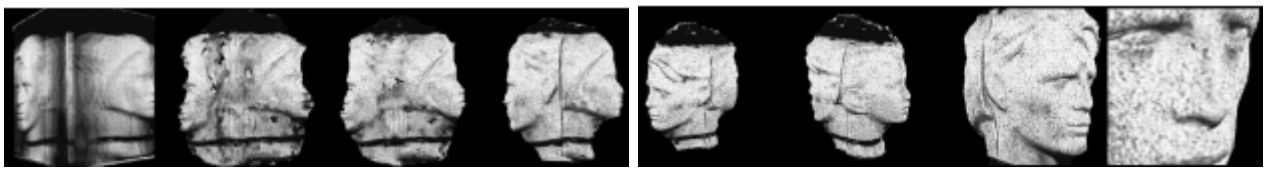


Figure 8. Reconstruction (left) and new view synthesis (right) of a two-headed object using the level set method. Images courtesy of Olivier Faugeras and Renaud Keriven.

by [Marr 76] who develops a relaxation network that enforces uniqueness and continuity constraints by introducing inhibitory and excitatory connections between voxels representing disparity hypotheses. [Yang 93] develops a multi-resolution method that operates in a fine-to-coarse manner to construct, then enhance, a disparity surface. A dynamic programming method is developed by [Intille 94] in a framework allowing explicit modeling of occlusions. A coarse-to-fine approach is presented by [Chen 99] who identifies seed voxels (those with strong evidence for a disparity solution) that are used to search for the global disparity surface.

Standard pair-wise matching methods are often limited for several reasons. First, input views can only be separated by a limited distance, or *baseline*, for correlation to be effective. Second, the result of pair-wise reconstruction is at best a 2½D reconstruction. Third, occlusion processes are difficult to model image space. Rather than approach the problem in image space, many successful techniques work instead in object space (i.e. 3D space). With a surface in 3D space, it is much easier to reason about occlusion relationships, as well as identify corresponding regions for correlation in image space. This is one of the reasons why voxel coloring is so effective. In the next subsection, we discuss a multi-view stereo vision technique that works in object space using level set theory.

4.2 Object Space Methods

Level set theory was developed by Osher and Sethian [Osher 88] to model the evolution of propagating interfaces. For 3D surface evolution, these methods start with an initial surface, which then moves with speed F along its normal. The goal is to track the evolution of the surface over time. Level set methods were initially developed for modeling flame propagation, but have since been applied to an astonishingly diverse array of problems [Sethian 99].

Level set methods embed this time-varying surface as the zero level set of a function $\phi(x, y, z, t)$. Level set theory provides an accurate and stable numerical scheme that solves the partial differential equations (PDEs) that characterize the motion for the surface. This scheme operates on a discrete 3D computational grid, i.e. a voxel space. At any time t_i, the surface can be extracted from the computational grid by identifying the zero level set, using Marching Cubes [Lorenson 87] to extract the isosurface where $\phi(x, y, z, t_i) = 0$.

[Faugeras 98] adapts the level set method to the scene reconstruction problem. In this approach, the initial surface is one that encompasses the scene. This initial surface is then evolved along its inwardly pointing normal, towards the objects in the scene. The speed of the zero level set slows as it approaches the true scene geometry, and attempts to lock onto it. During the evolution, the level set formulation can handle arbitrary topological changes, so the zero level set can break apart and merge if necessary.

Driving the surface evolution is the speed function, which is based on the cross-correlation of colors observed across pairs of views. When this cross-correlation is poor, the speed is high, which enables the zero level set to move through free space. However, as the zero level set gets near the 3D surfaces being reconstructed, the cross-correlation betters and the speed slows. The visibility of the zero level set is computed for each viewpoint, so that only the cameras that have an unoccluded view of a point on the zero level set contribute to the computation of the speed function of the point. This accurate handling of occlusion is provided by the object space approach. The results produced using this technique are impressive, and rival the best reconstructions achieved using voxel coloring methods. An example is provided in Figure 8.

This level set approach and voxel coloring approaches have many commonalities. First, both work on a dense voxel grid, and move an initial surface to the true scene geometry. Both of these methods use a color similarity measure to guide the reconstruction. Also, both use the correct visibility of the scene, and can account for arbitrary topological changes during reconstruction. These methods differ mainly in that the level set method was developed in an analytic framework in which the surface propagation is characterized by PDEs. This framework provides an analytic computation of the surface, as well as its intrinsic geometric properties, such as the normal vector and curvature.

5. CONCLUSION

We have presented a survey of methods for volumetric scene reconstruction, a topic that has received considerable interest in the past few years. We have discussed algorithms in three broad classes. The first class reconstructs a visual hull using geometric intersections, and easily reconstructs non-Lambertian scenes. The second class that we survey, voxel coloring, reconstructs a photo hull using color consistency measures. By taking advantage of the color information available in the images, voxel coloring methods can produce a reconstruction that is tighter fit to the true scene geometry than visual hull methods. Additionally, voxel coloring methods do not require that the images be segmentable into foreground / background regions. However, modeling non-Lambertian scenes becomes more difficult in this context.

Finally, we examine methods based on pair-wise matching. We look at both image space and object space approaches, the latter of which has advantages in determining occlusion relationships and regions for correlation matching. We discuss a level set method based on PDEs that model surface propagation.

Volumetric scene reconstruction has made significant progress over the last few decades, and many techniques have been proposed and refined. Future work in this field may include more sophisticated handling of non-Lambertian scenes, new methods for reconstruction of time-varying scenes, and more computationally efficient methods for real-time reconstruction.

6. REFERENCES

[Chen 99] Q. Chen and G. Medioni, "A Volumetric Stereo Matching Method: Application to Image-Based Modeling," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1999, pp. 29-34.

[Chien 84] C. H. Chien and J. K. Aggarwal, "A Volume/Surface Representation," *Proceedings of the International Conference on Pattern Recognition*, Montreal, Canada, July 30 – Aug. 2, 1984, pp. 817-820.

[Chien 86] C. H. Chien and J. K. Aggarwal, "Volume / Surface Octrees for the Representation of Three-Dimensional Objects," *Computer Vision, Graphics, and Image Processing*, Vol. 36, No. 1, Oct. 1986, pp. 100-113.

[Collins 96] R. Collins, "A Space-Sweep Approach to True Multi-Image Matching," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 18-20, 1996, pp. 358-363.

[Culbertson 99] W. B. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized Voxel Coloring," *Proceedings of the ICCV Workshop, Vision Algorithms Theory and Practice*, Springer-Verlag Lecture Notes in Computer Science 1883, September 1999, pp. 100-115.

[De Bonet 99] J. De Bonet and P. Viola, "Roxels: Responsibility Weighted 3D Volume Reconstruction," *Proceedings of the IEEE International Conference on Computer Vision,* 1999, Vol. 1, pp. 415-425.

[Eisert 99] P. Eisert, E. Steinbach, and B. Girod, "Multi-Hypothesis, Volumetric Reconstruction of 3-D Objects From Multiple Calibrated Camera Views," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing,* 1999, pp. 3509-3512.

[Faugeras 96] O. Faugeras, *Three-Dimensional Computer Vision, A Geometrical Viewpoint*, MIT Press, 1996.

[Faugeras 98] O. Faugeras and R. Keriven, "Variational Principles, Surface Evolution, PDE's, Level Set Methods, and the Stereo Problem," *IEEE Transactions on Image Processing*, Vol. 7, No. 3, March 1998, pp. 336-344.

[Fromherz 94] T. Fromherz and M. Bichsel, "Shape from Contours as Initial Step in Shape from Multiple Cues," *ISPRS Commission III Symposium on Spatial Information from Digital Photogrammetry and Computer Vision*, Munich, Germany, 1994, pp. 240-256.

[Fromherz 95] T. Fromherz and M. Bichsel, "Shape from Multiple Cues: Integrating Local Brightness Information," *Fourth International Conference for Young Computer Scientist, ICYCS 95*, Beijing, P. R. China, 1995, pp. 855-862.

[Fua 95] P. Fua and Y. G. LeClerc, "Object-Centered Surface Representations: Combining Multiple-Image Stereo and Shading," *International Journal of Computer Vision*, Vol. 16, No. 1, Sept. 1995, pp. 35-56.

[Garcia 98] B. Garcia and P. Brunet, "3D reconstruction with Projective Octrees and Epipolar Geometry," *Proceedings of the IEEE International Conference on Computer Vision*, Jan. 4-7, 1998, pp. 1067-1072.

[Gortler 96] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, "The Lumigraph," *Proceedings of A.C.M. SIGGRAPH*, 1996, pp. 43-54.

[Greenleaf 70] J. F. Greenleaf, T. S. Tu, and E. H. Wood, "Computer Generated 3-D Oscilloscopic Images and Associated Techniques for Display and Study of the Spatial Distribution of Pulmonary Blood Flow," *IEEE Transactions on Nuclear Science*, Vol. 17, No. 3, June 1970, pp. 353-359.

[Hartley 00] R. Hartley and A. Zisserman, *Multiple View Geometry*, Cambridge University Press, 2000.

[Kaufman 91] A. Kaufman, *Volume Visualization*, IEEE Computer Society Press, Los Alamitos, California, 1991.

[Kimura 99] M. Kimura, H. Saito, and T. Kanade, "3D Voxel Construction Based on Epipolar Geometry," *Proceedings of the International Conference on Image Processing*, 1999, pp. 135-139.

[Kutulakos 00a] K. N. Kutulakos, "Approximate N-View Stereo," *Proceedings of the European Conference on Computer Vision*, Springer Lecture Notes in Computer Science 1842, June/July 2000, Vol. 1, pp. 67-83.

[Kutulakos 00b] K. N. Kutulakos and S. M. Seitz, "A Theory of Shape by Space Carving," *International Journal of Computer Vision*, Vol. 38, No. 3, July 2000, pp. 199-218.

[Kutulakos 98] K. N. Kutulakos and S. M. Seitz, "What Do N Photographs Tell Us about 3D Shape?" *TR680*, Computer Science Dept. U. Rochester, January 1998.

[Laurentini 94] A. Laurentini, "The Visual Hull Concept for Silhouette-Based Image Understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2, Feb. 1994.

[Lorenson 87] W. Lorenson and H. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Proceedings of A.C.M. SIGGRAPH*, 1987, pp. 163-170.

[Luong 96] Q. Luong and O. Faugeras, "The Fundamental Matrix: Theory, Algorithms and Stability Analysis," *International Journal on Computer Vision*, Vol. 17, No. 1, Jan. 1996, pp. 43-75.

[Marr 76] D. Marr and T. Poggio, "Cooperative Computation of Stereo Disparity," *Science*, Vol. 194, No. 4262, Oct. 1976, pp. 283-287.

[Martin 83] W. Martin and J. K. Aggarwal, "Volumetric Descriptions of Objects from Multiple Views", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 5, No. 2, March 1983, pp. 150-158.

[Massone 85] L. Massone, P. Morasso, and R. Zaccaria, "Shape from Occluding Contours," *Proceedings of the SPIE Conference on Intelligent Robots and Computer Vision*, SPIE Vol. 521, Nov. 1985, pp. 114-120.

[Max 96] N. Max, X. Pueyo, and P. Schroder, "Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers," *Proceedings of the Eurographics Rendering Workshop*, 1996, pp. 165-174.

[Moezzi 96] S. Moezzi, A. Katkere, D. Kuramura, and R. Jain, "Reality Modeling and Visualization from Multiple Video Sequences," *IEEE Computer Graphics and Applications*, Vol. 16, No. 6, Nov. 1996, pp. 58-63.

[Moezzi 97] S. Moezzi, L. Tai, and P. Gerard, "Virtual View Generation for 3D Digital Video," *IEEE Multimedia*, Vol. 4, No. 1, Jan. - Mar. 1997, pp. 18-26.

[Osher 88] S. Osher and J. Sethian, "Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *Journal of Computational Physics*, 79, 1988, pp. 12-49.

[Pentland 90] A. Pentland, "Automatic Extraction of Deformable Part Models," *International Journal of Computer Vision*, Vol. 4, No. 2, March 1990, pp. 107-126.

[Potmesil 87] M. Potmesil, "Generating Octree Models of 3D Objects from Their Silhouettes in a Sequence of Images," *Computer Vision, Graphics and Image Processing*, Vol. 40, No. 1, Oct. 1987, pp. 1-29.

[Prock 98] A. Prock and C. Dyer, "Towards Real-Time Voxel Coloring," *Proceedings of the DARPA Image Understanding Workshop*, 1998, pp. 315-321.

[Saito 99] H. Saito and T. Kanade, "Shape Reconstruction in Projective Grid Space from Large Number of Images", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 23-25, 1999, Vol. 2, pp. 49-54.

[Seitz 95] S. Seitz and C. Dyer, "Complete Scene Structure from Four Point Correspondences," *Proceedings of the IEEE International Conference on Computer Vision*, June 1995, pp. 330-337.

[Seitz 97] S. Seitz and C. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1997, pp. 1067-1073.

[Seitz 99] S. Seitz and C. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," *International Journal of Computer Vision*, Vol. 35, No. 2, 1999, pp. 151-173.

[Sethian 99] J. Sethian, *Level Set Methods and Fast Marching Methods*, Cambridge University Press, Second Edition, 1999.

[Shade 98] J. Shade, S. Gortler, L. He, and R. Szeliski, "Layered Depth Images," *Proceedings of A.C.M. SIGGRAPH*, 1998, pp. 231-242.

[Shashua 95] A. Shashua and M. Werman, "On the Trilinear Tensor of Three Perspective Views and its Underlying Geometry," *Proceedings of the IEEE International Conference on Computer Vision*, 1995, pp. 920-925.

[Shneier 84] M. Shneier, E. Kent, and P. Mansbach, "Representing Workspace and Model Knowledge for a Robot with Mobile Sensors," *Proceedings of the International Conference on Pattern Recognition*, Montreal, Canada, July 1984, pp. 199-202.

[Slabaugh 00a] G. Slabaugh, T. Malzbender, and W. B. Culbertson, "Volumetric Warping for Voxel Coloring on an Infinite Domain," *Proceedings of the Workshop on 3D Structure from Multiple Images for Large-scale Environments (SMILE)*, July 2000, pp. 41-50.

[Slabaugh 00b] G. Slabaugh, W. B. Culbertson, T. Malzbender, and R. Schafer, "Improved Voxel Coloring Via Volumetric Optimization," *Center for Signal and Image Processing Technical Report TR3*, Georgia Institute of Technology, 2000.

[Srivastava 90] S. Srivastava and N. Ahuja, "Octree Generation from Object Silhouettes in Perspective Views," *Computer Vision, Graphics and Image Processing*, Vol. 49, No. 1, Jan. 1990, pp. 68-84.

[Steinbach 00a] E. Steinbach, B. Girod, P. Eisert, and A. Betz, "3-D Object Reconstruction Using Spatially Extended Voxels and Multi-Hypothesis Voxel Coloring," *Proceedings of the International Conference on Pattern Recognition*, 2000, Vol. 1, pp. 774-777.

[Steinbach 00b] E. Steinbach, B. Girod, P. Eisert, and A. Betz, "3-D Reconstruction of Real-World Objects Using Extended Voxels," *Proceedings of the International Conference on Image Processing*, 2000, Vol. III, pp. 138-141.

[Szeliski 93] R. Szeliski, "Rapid Octree Construction from Image Sequences," *Computer Vision, Graphics and Image Processing: Image Understanding*, Vol. 58, No. 1, July 1993, pp. 23-32.

[Szeliski 99] R. Szeliski and P. Golland, "Stereo Matching with Transparency and Matting," *International Journal of Computer Vision*, Vol. 32, No. 1, 1999, pp. 45-62.

[Terzopoulos 91] D. Terzopoulos and D. Metaxas, "Dynamic 3D Models with Local and Global Deformations: Deformable Superquadrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, No. 7, July 1991, pp. 703-714.

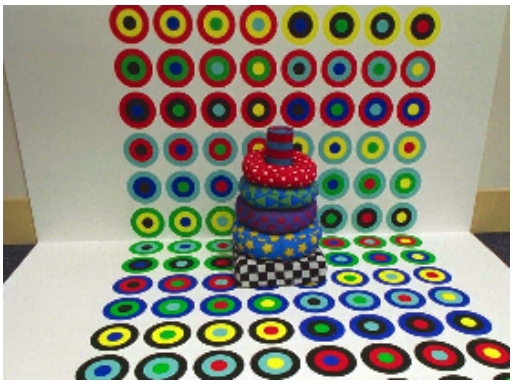
[Tsai 87] R. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE Transactions on Robotics and Automation*, Vol. 3, No. 4, Aug. 1987, pp. 323-344.

[Vedula 00] S. Vedula, S. Baker, S. Seitz, and T. Kanade, "Shape and Motion Carving in 6D," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000, Vol. 2, pp. 592-598.

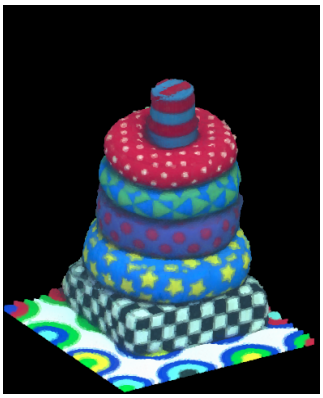
[Veenstra 86] J. Veenstra and N. Ahuja, "Efficient Octree Generation from Silhouettes," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, June 1986, pp. 537-542.

[Weghorst 84] H. Weghorst, G. Hooper, D. P. Greenberg, "Improving Computational Methods for Ray Tracing", *ACM Transactions on Graphics*, Vol. 3, No. 1, January 1984, pp. 52-69.

[Yang 93] Y.Yang, A. Yuille, and J. Lu, "Local, Global, and Multilevel Stereo Matching," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1993, pp. 274-279.



(a)



(b)

(c)

(d)

Figure 9. Scene reconstruction and new view synthesis. Reference views are shown in (a) and (c), new views synthesized from the reconstruction are shown in (b) and (d). The Voxel Coloring algorithm was used to produce (b). GVC-IB was used to produce (d).

Figure 10. On the left is one of ten panoramic photographs used in a reconstruction using a warped voxel space. On the right is a new view rendered from the reconstruction.