Methods for Volumetric Reconstruction of Visual Scenes

GREGORY G. SLABAUGH
Intelligent Vision and Reasoning Department, Siemens Corporate Research, Princeton, NJ 08540
greg.slabaugh@scr.siemens.com

W. BRUCE CULBERTSON, THOMAS MALZBENDER
Visual Computing Department, Hewlett-Packard Laboratories, Palo Alto, CA 94304
{bruce_culbertson, tom_malzbender}@hp.com

MARK R. STEVENS
Charles River Analytics Inc., Cambridge, MA 02138
mstevens@cra.com

RONALD W. SCHAFER
Center for Signal and Image Processing, Georgia Institute of Technology, Atlanta, GA 30318
rws@ece.gatech.edu

Abstract

In this paper, we present methods for 3D volumetric reconstruction of visual scenes photographed by multiple calibrated cameras placed at arbitrary viewpoints. Our goal is to generate a 3D model that can be rendered to synthesize new photo-realistic views of the scene. We improve upon existing voxel coloring / space carving approaches by introducing new ways to compute visibility and photo-consistency, as well as model infinitely large scenes. In particular, we describe a visibility approach that uses all possible color information from the photographs during reconstruction, photo-consistency measures that are more robust and/or require less manual intervention, and a volumetric warping method for application of these reconstruction methods to large-scale scenes.

Keywords

Scene reconstruction, voxel coloring, space carving, photo-consistency, histogram intersection, volumetric warping

1 Introduction

In this paper, we consider the problem of reconstructing a 3D model of a scene of unknown geometric structure using a set of photographs (also called reference views) of the scene taken from calibrated and arbitrarily placed cameras. Our goal is to reconstruct geometrically complex scenes using a set of easily obtained photographs taken with inexpensive digital cameras. We then project this reconstructed 3D model to virtual viewpoints in order to synthesize new views of the scene, as shown in Figure 1.

To accomplish this task, we have developed methods that improve upon the quality, usability, and applicability of existing volumetric scene reconstruction approaches. We present innovations in the computation of visibility and photo-consistency, which are two crucial aspects of this class of algorithms. One of our visibility approaches minimizes photo-consistency evaluations, which results in efficient computation, and our histogram intersection method of computing photo-consistency requires almost no user intervention. We also present a volumetric warping approach designed to reconstruct infinitely large scenes using a finite number of voxels. These techniques are aimed at bringing volumetric scene reconstruction out of the laboratory and towards the reconstruction of complex, real-world scenes.

1.1 Related Work

The 3D scene reconstruction problem has received considerable attention in the literature, and a multitude of solutions have been proposed. Many solutions have been developed for specific camera configurations (e.g., a small number of cameras [28], short baselines, parallel optical axes [4], an ordinal visibility constraint [39], etc.); or specific classes of scenes (e.g., scenes composed of geometric primitives such as planes [15, 21], lines, or curves, scenes exhibiting or lacking texture [52], etc.). Some solutions re-

(a) (b)

Figure 1: One of 24 reference views of our "Ceevah" data set (a) and a new view synthesized after scene reconstruction (b).

quire user interaction [11, 41]. In this paper, we are interested in a more general case, for which a scene of unknown geometric structure is photographed from any number of arbitrarily placed cameras. We are most interested in techniques that require minimal interaction with the user.

In the literature, several methods to represent a visual scene have been proposed, including layered depth images [40], surface meshes, surfels [6, 34], light fields [18, 26], etc. In this paper, we focus on volumetric representations, which provide a topologically flexible way to characterize a 3D surface inferred from multiple images. If desired, a voxel-based surface can be converted into any of the above representations with relative ease.

Due to the large number of scene reconstruction approaches, it would be impossible to provide a comprehensive review here; see [12, 43] for a survey of volumetric approaches. Techniques such as multi-view stereo [17, 32] and structure from motion [1, 20, 35] have been quite successful at reconstructing 3D scenes. These methods compute and then triangulate correspondences between views to yield a set of 3D points that are then fit to a surface. The effectiveness of these reconstruction methodologies relies upon accurate image-space correspondence matching. Such matching typically falters as the baseline between views increases since the effects of occlusion and perspective are difficult to model in image space when the scene geometry is unknown. Consequently, many of these methods are not well suited to the arbitrary placement of cameras.

A level set approach to the scene reconstruction problem has been proposed by Faugeras and Keriven [16]. A surface initially larger than the scene is evolved using partial differential equations to a successively better approximation of true scene geometry. Like the approaches we present in this paper, this level set method can employ arbitrary numbers of images, account for occlusion correctly, and deduce arbitrary topologies.

Perhaps the simplest class of volumetric multi-view reconstruction methods are visual hull approaches [25, 29, 47]. The visual hull, computed from silhouette images, is an outer-bound approximation to the scene geometry. Algorithms that compute the visual hull are applicable to scenes with arbitrary BRDFs as long as foreground/background segmentation at each reference view is possible, and are relatively simple to implement since visibility need not be modeled when reconstructing the scene geometry.

While a visual hull can be rendered to produce new views of the scene, typically the visual hull geometry is not very accurate. This can diminish the photo-realism when new views are synthesized. To increase the geometric accuracy, more information than silhouettes must be used during reconstruction. Color is an obvious source of such additional information. Many researchers have attempted to reconstruct 3D scenes by analyzing colors across multiple viewpoints. Specifically, they have sought a 3D model that, when projected to the reference views, reproduces the photographs.

Reconstructing such a model requires a photo-consistency check, which determines if a point in 3D space is consistent with the photographs taken of the scene. In particular, a point is photo-consistent [39, 24] if:

- It does not project to background, if the background is known.
- When the point is visible, the light exiting the point (i.e., radiance) in the direction of the camera is equal to the observed color of the point's projection in the

photograph.

Kutulakos and Seitz [24] state that surfaces that are not transparent or mirror-like reflect light in a coherent manner; that the color of light reflected from a single point along different directions is not arbitrary. The photo-consistency check takes advantage of this fact to eliminate visible parts of space that do not contain scene surfaces.

This reconstruction problem is ill-posed in that, given a set of photographs and a photo-consistency check, there are typically multiple 3D models that consist of photo-consistent points. In their insightful work, Kutulakos and Seitz [24] introduce the photo hull, which is the largest shape that contains all reconstructions in the equivalence class of photo-consistent 3D models. For a given monotonic photo-consistency check [1], the photo hull is unique, and is itself a reconstruction of the scene. Since we model points with voxels, the photo hull is found by identifying the spatially largest volume of voxels that are photo-consistent with all reference views.

When computing the photo hull, we have found that the quality of the result depends heavily on two factors. They are:

1. Visibility: The method of determining of the pixels from which a voxel V is visible. We denote these pixels π^V.

2. Photo-consistency test: A function that decides, based on π^V, whether a surface exists at V.

In the algorithm presented in the next paragraph, we will see that visibility and photo-consistency are inter-related and, as a result, multiple passes must in general be made over the voxels to find the photo hull.

Volumetric methods for finding the photo hull adopt the following approach. First, a voxel space is defined that contains, by a comfortable margin, the portion of the scene to be reconstructed. During reconstruction, the voxels are either completely transparent or opaque; initially, they are all opaque. Voxels that are visible to the cameras are checked for photo-consistency, and the inconsistent voxels are carved, i.e., their opacity is set to transparent. Carving one voxel typically changes the visibility of other opaque voxels. Since the photo-consistency of a voxel is a function of its visibility, the consistency of an uncarved voxel must be rechecked whenever its visibility changes. The algorithm continues until all visible, uncarved voxels are photo-consistent. This set of voxels, when rendered to the reference views, reproduces the photographs and is therefore a model that resembles the scene. Pseudocode is provided in Figure 2.

[1] We will discuss monotonicity in Section 2.

```
set all voxels uncarved
loop {
  for every uncarved voxel V {
    find π^V
    if (π^V is inconsistent)
      carve V
  }
  if (no voxels carved on this iteration)
    done
}
```

Figure 2: Generic pseudocode for reconstructing the photo hull.

The Voxel Coloring algorithm of Seitz and Dyer [39] reconstructs the photo hull for scenes photographed by cameras that satisfy the *ordinal visibility constraint*, which restricts the camera placements so that the voxels can be visited in an order that is simultaneously near-to-far relative to every camera. Typically, this condition is met by placing all the cameras on one side of the voxel space, and processing voxels using plane that sweeps through the volume in a direction away from the cameras. Under this constraint, visibility is simple to model using occlusion bitmaps [39].

Voxel Coloring is elegant and efficient, but the ordinal visibility constraint is a significant limitation, since it means that cameras cannot surround the scene. Kutulakos and Seitz [23] present what we call the Partial Visibility Space Carving (PVSC) algorithm, which repeatedly sweeps a plane through the volume in all six axis-aligned directions. For each plane sweep, only the subset of cameras that are behind the plane are used in the photo-consistency check. This approach permits arbitrary camera placement, which is a significant advantage over Voxel Coloring. However, when evaluating a voxel's photo-consistency, it uses pixels from only a subset of the total cameras that have visibility of the voxel. To address this issue, Kutulakos and Seitz [24] subsequently include some additional per-voxel bookkeeping that accumulates the visible pixels in the voxel's projection as the plane is swept in all six axis-aligned directions. On the sixth sweep, the full visibility of the voxel is known and considered in the photo-consistency check. We call this version of their algorithm Full Visibility Space Carving (FVSC).

Such carving algorithms are quite powerful, and have captured the interest of many researchers who have proposed extensions to or reformulations of the basic approach. Briefly, Prock and Dyer [36] present a multi-resolution approach as well as hardware implementations for improved efficiency. Researchers have performed space carving using intrinsically calibrated [14] and weakly calibrated [22, 37] cameras. Space carving was recast in probabilistic frameworks by Broadhurst et al. [5] and Bhotika et

al. [2]. Researchers have developed carving algorithms for scenes with shadows [38], opacity [10], mixed pixels [48], and non-Lambertian surfaces [6, 7]. Vedula et al. [50] and Carceroni and Kutulakos [6] propose carving algorithms for reconstructing time-varying scenes. Slabaugh et al. [44] present an epipolar approach to constructing view-dependent photo hulls at interactive rates.

1.2 Contributions

This paper presents contributions in three areas; visibility, photo-consistency, and the modeling of infinitely large scenes. We discuss each below.

As stated above, visibility is a vital part of any algorithm that reconstructs the photo hull. In Section 2 we present a scene reconstruction approach, Generalized Voxel Coloring (GVC), which introduces novel methods for computing visibility during reconstruction. These methods support arbitrary camera placement and place minimal requirements on the order in which voxels are processed, unlike plane sweep methods [39, 24]. We show that one of our new methods minimizes photo-consistency checks. We also demonstrate how full visibility can result in more accurate reconstructions for real-world scenes.

The photo-consistency test is the other crucial part of an algorithm that reconstructs the photo hull. In Section 3 we introduce two novel photo-consistency tests for Lambertian scenes. The first is an adaptive technique that adjusts the photo-consistency test so that surface edges and textured surfaces can be more accurately reconstructed. The second is based on color histograms and treats multi-modal color distributions in a more principled way that earlier approaches. In addition, our histogram-based photo-consistency test requires little parameter tuning.

Reconstruction of large-scale scenes that contain objects both near to and far from the cameras is a challenging problem. Modeling such a scene with a fixed resolution voxel space is often inadequate. Using a high enough resolution for the foreground may result in an unwieldy number of voxels that becomes prohibitive to process. Using a lower resolution, more suitable to the background, may result in an insufficient resolution for the foreground. In Section 4 we present a volumetric warping approach that represents infinitely large scenes with a finite number of voxels. This method simultaneously models foreground objects, background objects, and everything in between, using a voxel space with variable resolution. Using such a voxel space in conjunction with our GVC approach, we reconstruct and synthesize new views of a large outdoor scene.

We note that some of the content of this paper has appeared in previous workshop and conference papers [9, 42, 46].

2 Generalized Voxel Coloring

We present two closely related space carving algorithms [2] that we collectively call Generalized Voxel Carving (GVC). They differ from each other, and from earlier space carving algorithms, primarily in the means they compute visibility. The earlier methods require the voxels to be scanned in plane sweeps, whereas GVC scans voxels in a more general order. GVC represents just visible surface voxels in its main data structure, reducing both computation and storage. GVC accounts for visibility in a way that very naturally accommodates arbitrary camera placement and allows full voxel visibility to be computed efficiently. The first GVC algorithm, GVC-IB, uses less memory than the other. It also uses incomplete visibility information during much of the reconstruction yet, in the end, computes the photo hull using full visibility. The other GVC algorithm, GVC-LDI, uses full visibility at all times, which greatly reduces the number of photo-consistency checks required to produce the photo hull. We show that the use of full visibility results in better reconstructions than those produced by earlier algorithms that only use partial visibility.

As mentioned earlier, carving one voxel can change the visibility of other voxels, so visibility must be calculated frequently while reconstructing a photo hull. Because of self-occlusion in the scene, visibility is complex and potentially costly to compute. Thus, an efficient means of computing visibility is a key element of any practical space carving algorithm, including our GVC algorithms.

As a space carving algorithm begins a reconstruction, it carves voxels based on the visibility of a model that looks nothing like the final photo hull. One might therefore wonder: could the algorithm carve a voxel that belongs in the photo hull? To answer this question, consider the following two insights, based on Seitz and Dyer [39]. First, since voxels change from opaque to transparent during reconstruction, and never the reverse, the visibility of the remaining voxels can only increase. In particular, if S is the set of pixels that have an unoccluded view of an uncarved voxel at one point in time and if S' is the set of such pixels at a later point in time, then $S \subseteq S'$. Second, Seitz and Dyer make an unstated assumption that the consistency test is *monotonic*, meaning for any two sets of pixels S and S' with $S \subseteq S'$, if S is inconsistent, then S' is also inconsistent. These two facts imply that carving is *conservative*: no voxel will ever be carved if it would be photo-consistent in the final model. (Although carving with non-monotonic consistency tests is not in general conservative, we show in

[2] Throughout this paper, we will use the term "space carving algorithms" to refer to the class of volumetric scene reconstruction algorithms that use photo-consistency to carve a voxel space. Voxel Coloring, PVSC, FVSC, and GVC are all members of this class.

Section 3 that such tests can nevertheless yield good looking reconstructions.)

Both GVC-IB and GVC-LDI maintain a surface voxel list (SVL), a list of the surface voxels in the current model that are visible from the cameras. For box-shaped voxel spaces that contain none of the cameras, we typically initialize the SVL with the outside layer of voxels. We have used ad hoc methods to initialize the SVL when we used more complicated voxel spaces, as in Section 4. When a voxel is carved, we remove it from the SVL. We also add to the SVL any voxels that are adjacent to the carved voxel and that have not been previously carved; this prevents holes from being introduced into the surface represented by the SVL. As described below, we give each voxel a unique ID number. We use a hash table to find the voxel with a given ID in the SVL. The SVL can also be scanned sequentially.

2.1 The GVC-IB Algorithm

The GVC-IB algorithm maintains visibility using an item buffer [51] for each reference image. An item buffer is defined as follows: for each pixel P in a reference image, an item buffer, shown in Figure 3a, stores the voxel ID of the closest voxel that projects to P (if any). An item buffer is computed by rendering the voxels into the reference image using z-buffering, but storing voxel IDs instead of colors. As with earlier space carving algorithms, it is assumed that at most one voxel is visible from any pixel. Therefore, we make no attempt to model blended colors that arise from transparency [10] or depth discontinuities [48].

Pseudocode for GVC-IB appears in Figure 4. Once valid item buffers have been computed for the images, their pixels are then scanned. During the scan, if a valid voxel ID is found in a pixel's item buffer value, then the pixel's color is accumulated into the voxel's color statistics.

When the pixel scanning is complete, the SVL is scanned and each voxel is tested for consistency, based on the collected color statistics. If a voxel is found to be inconsistent, it is carved and removed from the SVL. After a voxel is carved, the visibility of the remaining SVL voxels potentially changes, so all the color statistics must be considered out-of-date. At this point, it might seem natural to recompute the item buffers and start the process all over. However, because the item buffers are time-consuming to compute, we delay updating them. Although the visibility found using out-of-date item buffers is no longer valid for the current model, it is still valid for a superset of the current model. Because carving is conservative, no consistent voxels will be carved using the out-of-date color statistics, though some voxels that should be carved might not be. When the entire SVL has been scanned and all voxels with inconsistent color statistics have been carved, then we recompute the item buffers and begin again. These iterations continue until, during some iteration, no carving occurs. At this point, the SVL is entirely consistent, based on up-to-date visibility, so the SVL is in fact the photo hull.

```
initialize SVL
loop {
  for all images
    compute item buffer
  accumulate color statistics into SVL voxels
  for every voxel V in SVL {
    if (V is inconsistent) {
      carve V (remove V from SVL)
      add uncarved neighbors of V to SVL
    }
  }
  if (no voxels carved)
    done
}
```

Figure 4: GVC-IB pseudocode.

Profiling GVC-IB revealed that nearly all the runtime is spent rendering item buffers. This suggested two ways to accelerate the algorithm. Since each item buffer is independent of the others, they can be rendered in parallel on a multi-CPU computer. Using two CPUs and several image sets, we measured runtime reductions between 46% and 48% compared to a uniprocessor. Next, we tried rendering the item buffers with a hardware graphics accelerator. This resulted in runtime reductions between 56% and 63%. GVC-IB (and GVC-LDI) can also be executed in a coarse-to-fine manner, as described in [36]. We have seen runtime reductions of approximately 50% using such multi-resolution voxel spaces. These efficiencies can be combined for faster reconstructions.

2.2 The GVC-LDI Algorithm

GVC-IB computes visibility in a relatively simple manner that also makes efficient use of memory. However, the visibility information is time consuming to update. Hence, GVC-IB updates it infrequently and it is out-of-date much of the time. Using a monotonic photo-consistency measure, this does not lead to incorrect results but it does result in inefficiency because a voxel that would be evaluated as inconsistent using all the visibility information might be evaluated as consistent using a subset of the information. Ultimately, all the information is collected but, in the meantime, voxels can remain uncarved longer than necessary and can therefore require more than an ideal number of consistency evaluations. Furthermore, GVC-IB reevaluates the consistency of voxels on the SVL even when their visibility (and hence their consistency) has not changed since their last evaluation. By using layered depth images

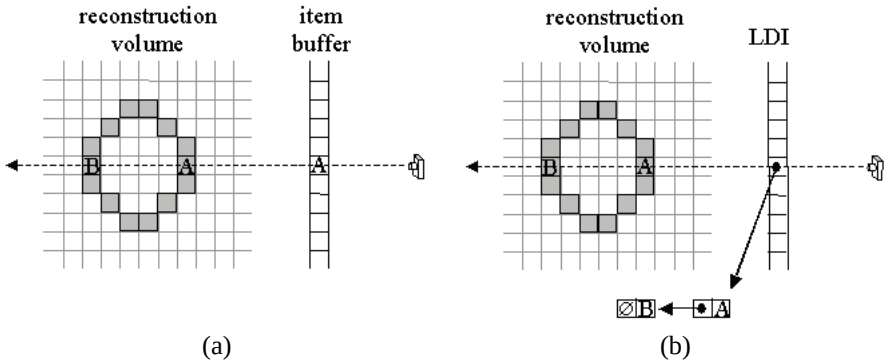


Figure 3: The data structures used to compute visibility. An item buffer (a) is used by GVC-IB and records the ID of the surface voxel visible from each pixel in an image. A layered depth image (LDI) (b) is used by GVC-LDI and records all surface voxels that project onto each pixel.

instead of item buffers, GVC-LDI can efficiently and immediately update the visibility information when a voxel is carved and also can precisely determine the voxels whose visibility has changed.

Unlike the item buffers used by the GVC-IB method, which record at each pixel P just the closest voxel that projects onto P, the LDIs store at each pixel a list of all the surface voxels that project onto P. See Figure 3b. These lists are sorted according to the distance from the voxel to the image's camera. The head of an LDI list stores the voxel closest to P, which is the same voxel an item buffer would store. The LDIs are initialized by rendering the SVL voxels into them.

Using the LDIs, the set of pixels π^V from which a voxel V is visible can be found as follows. V is scan converted into each reference image to find its projection (without regard to visibility). For each pixel P in V's projection, if the voxel ID at the head of P's LDI equals V's ID, then P is added to π^V. Once π^V is computed, V's consistency can be determined by testing π^V.

The uncarved voxels whose visibility changes when another voxel is carved come from two sources:

- They are inner voxels adjacent to the carved voxel and become surface voxels when the carved voxel becomes transparent. See Figure 5a.
- They are already surface voxels (hence they are in the SVL and LDIs) and are often distant from the carved voxel. See Figure 5b.

Voxels in the first category are trivial to identify since they are next to the carved voxel. Voxels in the second category are impossible to identify efficiently in the GVC-IB method; hence, that method must repeatedly evaluate the entire SVL for color consistency. In GVC-LDI, voxels in the second category can be found easily with the aid of the LDIs; they will be the second voxel on the LDI list for some pixel in the projection of the carved voxel. GVC-LDI keeps a list of the SVL voxels whose visibility has changed, called the changed visibility SVL (CVSVL). These are the only voxels whose consistency must be checked. Carving is finished, and the photo hull is found, when the CVSVL is empty.

When a voxel is carved, the LDIs (and hence the visibility information) can be updated immediately and efficiently. The carved voxel can be easily deleted from the LDI list for every pixel in the voxel's projection. The same process automatically updates the visibility information for the second category of uncarved voxels whose visibility has changed; these voxels move to the head of LDI lists from which the carved voxel has been removed and they are also added to the CVSVL. Inner voxels adjacent to the carved voxel are pushed onto the LDI lists for pixels they project onto. As a byproduct of this process, the algorithm learns if the voxel is visible; if it is, it is put on the CVSVL. Pseudocode for GVC-LDI is given in Figure 6.

2.3 GVC Reconstruction Results

We now present experimental results to demonstrate our GVC algorithms, and provide, for side-by-side comparison, results obtained with Space Carving. As discussed in Section 1.1, there are two versions of the Space Carving algorithm: Partial Visibility Space Carving (PVSC) and Full Visibility Space Carving (FVSC). As will be shown, PVSC produces less accurate results than GVC and FVSC. Therefore, we will focus more on comparing GVC to FVSC.

2.3.1 Comparison with Partial Visibility Space Carving

Figure 7 shows two of fifteen reference views of our "bench" data set. Calibration of the 765 x 509 pixel images

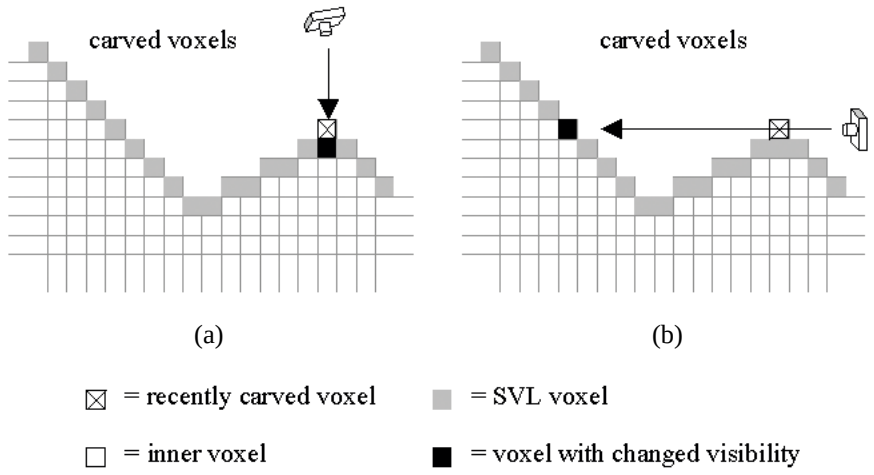


Figure 5: When a voxel is carved, there are two categories of other voxels whose visibility changes: (a) inner voxels that are adjacent to the carved voxel and (b) voxels that are already on the SVL and are often distant from the carved voxel.

```
initialize SVL
for all images
  compute LDI
place all voxels in SVL onto CVSVL
while (CVSVL not empty) {
  choose a voxel V from CVSVL
  remove V from CVSVL
  scan convert V to find π^V
  if (π^V is not consistent) {
    carve V (remove from SVL, LDIs)
    for all inner neighbors U of V
      add U to SVL, CVSVL, LDIs
    for voxels U that move to head of LDIs
      add U to CVSVL
  }
}
```

Figure 6: GVC-LDI pseudocode.

had accuracy of a maximum 1.2 pixels of reprojection error for the points used in the calibration. We reconstructed the scene using a 75 x 71 x 33 voxel volume. New views synthesized from the GVC-IB and PVSC reconstructions are shown in Figure 8.

The PVSC image is considerably noisier and more distorted than the GVC image, a trend we observed with all data sets we tested. In general, PVSC produces less accurate reconstructions than GVC, since, when computing photo-consistency, PVSC does not use the full visibility of the scene, unlike GVC. During a plane sweep, the cameras that are ahead of the plane are not considered by the PVSC algorithm even though those cameras might have visibility of voxels on the plane. Since photo-consistency is determined using a subset of the available color information, the photo-consistency test sometimes fails to produce the proper result had the full visibility been considered. For some data sets, we found the PVSC runs faster, while for others, GVC runs faster. However, PVSC always requires less memory than GVC-IB or GVC-LDI.

Additional comparisons between GVC and PVSC appear in [9].

2.3.2 Comparison with Full Visibility Space Carving

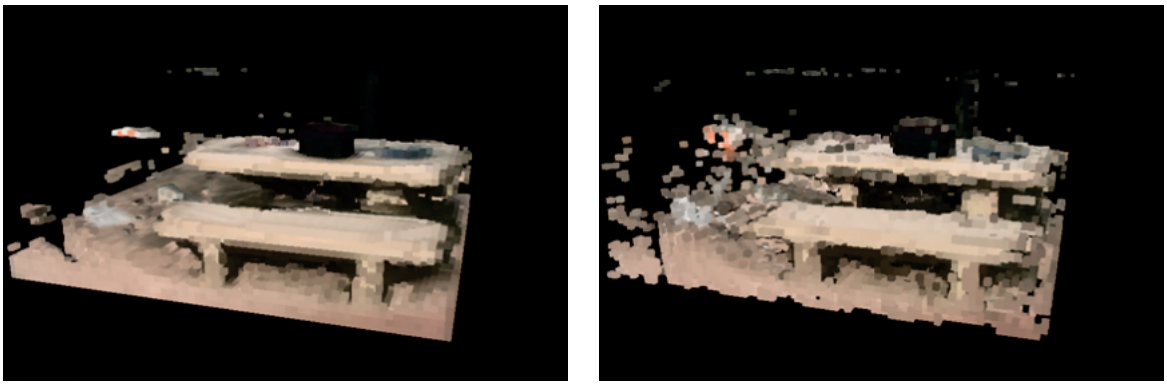
Next, we present results of running GVC-IB, GVC-LDI, and Full Visibility Space Carving (FVSC) on two data sets we call “toycar” and “ghirardelli”. In particular, we present runtime statistics and provide images synthesized with FVSC and our algorithms. The experiments were run on a computer with a 1.5 GHz Pentium 4 processor and 768 MB of RAM.

The toycar and ghirardelli data sets are quite different in terms of how difficult they are to reconstruct. The toycar scene is ideal for reconstruction. The seventeen 800 x 600 pixel images are computer-rendered and perfectly calibrated. The colors and textures make the various surfaces in the scene easy to distinguish from each other. Two of our toycar reference views are shown in Figure 9. In contrast, the seventeen 1152 x 872 ghirardelli images are imperfectly calibrated photographs of an object that has significant areas with relatively little texture and color variation. Two of our ghirardelli reference views are shown in Figure 11.

We reconstructed the toycar scene in a 167 x 121 x 101

(a) (b)

Figure 7: Two of the fifteen input images of the bench scene.



(a) (b)

Figure 8: New views projected from reconstructions of the bench scene. The image on the left was created with GVC-IB. The image on the right was created with Partial Visibility Space Carving. The PVSC image is considerably noisier and more distorted than the GVC-IB image.

voxel volume. The reconstruction of the ghirardelli data set occurred in a 168 x 104 x 256 voxel volume; note this resolution is significantly higher than that used to reconstruct the toycar scene. New views synthesized from reconstructions obtained using the GVC-IB, GVC-LDI, and FVSC algorithms are shown in Figures 10 and 12 for the toycar and ghirardelli data sets, respectively. The three reconstructions in each figure are not identical because we used a photo-consistency test (the adaptive standard deviation test that will be discussed in Section 3.2.1) that is not monotonic. Therefore, the order in which the voxels were processed affected the final result. However, for each data set, the three reconstructions are comparable in terms of quality.

There were significant differences between the algorithms in terms of runtime statistics, as shown in Table 1. The "Checks" column in the table indicates the number of photo-consistency checks that were required to complete the reconstruction. For both data sets, FVSC required an order of magnitude more consistency checks than the GVC algorithms, for two reasons. First, on each sweep through the volume, FVSC processes inner voxels, i.e., voxels that are inside the surface and not visible to any of the cameras. GVC, in contrast, does not process the inner voxels, instead only processing surface voxels. GVC-LDI is particularly efficient, since it only processes the surface voxels that change visibility, resulting in a minimal number of photo-consistency checks. Second, some voxels can only be carved using a large amount (possibly all) of the their total visibility. In FVSC, the amount of visibility grows from nothing at the beginning of the first sweep through the volume, to full visibility at the end of the sixth sweep. During some of these earlier sweeps, there may not be enough visibility for the voxel to be carved. In contrast, GVC uses

(a) (b)

Figure 9: Two of the seventeen images of the toycar scene.

(a) (b) (c)

Figure 10: New views of the toycar scene generated by rendering the GVC-IB (a), GVC-LDI (b), and FVSC (c) reconstructions.

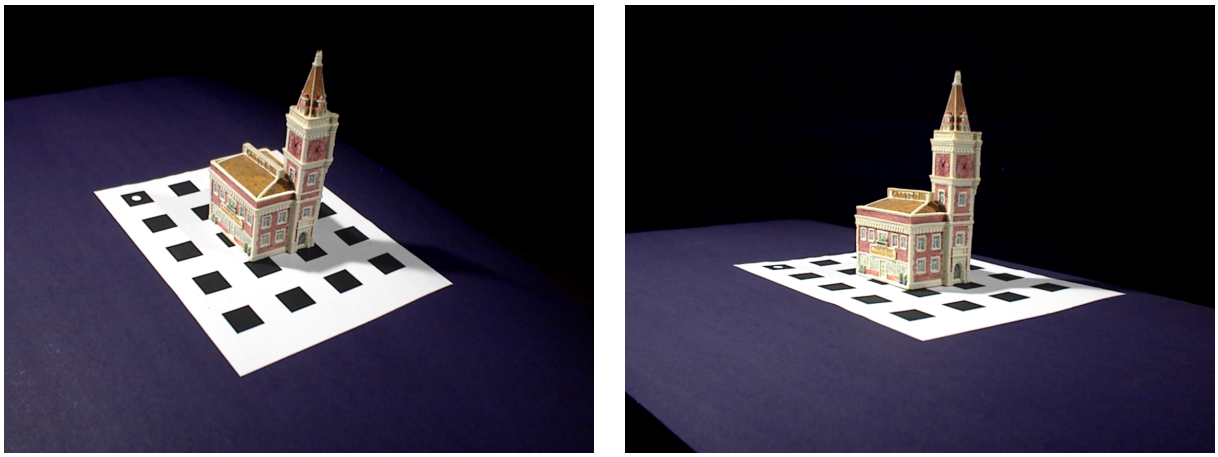
Data Set	Algorithm	Checks	Time (m:s)	Memory
Toycar	FVSC	25.8 M	32:31	156 MB
Toycar	GVC-IB	3.1 M	36:16	74 MB
Toycar	GVC-LDI	2.2 M	29:16	399 MB
Ghir.	FVSC	154 M	2:35:43	337 MB
Ghir.	GVC-IB	12.1 M	2:01:27	154 MB
Ghir.	GVC-LDI	4.5 M	0:47:10	275 MB

Table 1: Runtime statistics for the toycar and ghirardelli data sets.

more visibility. In particular, GVC-LDI always uses full visibility each time a voxel's photo-consistency is checked.

The "Time" column in Table 1 indicates the amount of time required to complete the reconstruction. For the toycar data set, the GVC algorithms were slightly faster than FVSC. Although GVC processes fewer voxels, the additional overhead required to maintain the visibility data structures does not result in a significantly faster runtime. However, for the Ghirardelli data set, the efficiency of GVC-LDI's relatively complex data structures more than compensates for the time needed to maintain them. Because GVC-LDI finds all the pixels from which a voxel is visible, it can carve many voxels sooner, when the model is less refined, than GVC-IB. Furthermore, after carving a voxel, GVC-LDI only reevaluates the few other voxels whose visibility has changed. Consequently, GVC-LDI is faster than GVC-IB by a large margin. For FVSC, the large number of photo-consistency checks results in a slower runtime.

The last column of Table 1 shows the memory usage of the algorithms. All three approaches keep copies of the



(a) (b)

Figure 11: Two of the seventeen reference views of the ghirardelli scene.

(a) (b) (c)

Figure 12: New views of the ghirardelli scene generated by rendering the GVC-IB (a), GVC-LDI (b), and FVSC (c) reconstructions.

input images in memory. As described in [24], for Lambertian scenes, FVSC additionally stores color statistics for each voxel in the voxel space. This grows as $O(N^3)$, where N is the number of voxels along one dimension of the voxel space. Additionally, we store the one of six partitions [24] of space that each camera lies in for each voxel. This storage is also $O(N^3)$. We note that the partitions could be computed on the fly (i.e., requiring no storage) at the expense of runtime. However, in our experiments, we opted for a faster runtime. Unlike FVSC, the voxel resolution has little bearing on the memory requirements for GVC-IB and GVC-LDI. GVC-IB requires equal amount of memory for the images and the item buffers. The LDIs dominate the memory usage in GVC-LDI and consume an amount of memory roughly proportional to the number of image pixels times the depth complexity of the scene. The SVL and CVSVL data structures used by GVC-IB and GVC-LDI require $O(N^2)$ storage, and are relatively insignificant. Thus, of the three approaches, GVC-IB required the least amount of memory. For the toycar data set modeled with a lower resolution voxel space, GVC-LDI required more memory than FVSC. However, for the ghirardelli data set modeled with a higher resolution voxel space, GVC-LDI required less memory than FVSC.

2.4 Summary

In this section we presented our Generalized Voxel Coloring algorithms, GVC-IB and GVC-LDI. These approaches support arbitrary camera placement and reconstruct the scene using full visibility. We demonstrated that methods like GVC that use full visibility result in more accurate reconstructions than those that use partial visibility. The GVC-IB algorithm is memory efficient, while our GVC-LDI algorithm reconstructs the scene using a minimal number of photo-consistency checks, which, for many scenes, results in a faster reconstruction.

3 Photo-Consistency Tests

When reconstructing a scene using a space carving algorithm, there are two key factors that affect the quality of the reconstructed model. The first is the visibility that is computed for the voxels. In the previous section we demonstrated that using full visibility produces better quality reconstructions than using only partial visibility. The second factor is the test that is used to judge the photo-consistency of voxels.

The section begins by describing the likelihood ratio test, the first consistency test that was proposed for space carving. We then describe several of the most straightforward, and perhaps obvious, candidate tests. Next, we present two tests that we have developed, the adaptive standard deviation test and the histogram test. These two tests have consistently yielded the best results in our new view synthesis application, and one of the tests has the added advantage of requiring little or no parameter adjustment. Finally, we present results that show some color spaces are better than others for space carving.

We have provided Figure 13 for comparison of the consistency tests and color spaces. It shows reconstructions performed with identical programs, aside from the tests or color spaces being compared. All the reconstructions in the figure use the same "shoes" data set, consisting of 30 1536×1024 images, but the results are consistent with other data sets we have tried. The reconstructions have been rendered to an identical viewpoint that is different from any of the input images used in the reconstructions. Parameters used in the tests were tuned to minimize holes in the calibration target that serves as the floor of the scene. Oğuz Özün [33] has also compared consistency tests and had similar success with the two tests we developed.

Kutulakos and Seitz [24] have stated that the photo hull, the set of all photo consistent voxels, provides "the tightest possible bound on the shape of the true scene that can be inferred from N photographs". However, different photo consistency tests lead to different photo hulls, many of which do not resemble the scene. If there is a voxel that belongs in a reconstruction but is judged by the test to be inconsistent, then space carving carves the voxel from the model. Worse, because the voxel is then considered transparent, the algorithm can draw incorrect conclusions about which images see the remaining uncarved voxels, leading to more incorrect carving. Figure 15b shows an example of this problem. The consistency test just described can be thought of as being too strict for declaring voxels that belong in the model to be inconsistent. Tests can also be too lenient, declaring voxels to be consistent when they do not belong in the model; this can lead to voxels that appear to float over a reconstructed scene. A single consistency test can simultaneously be both too strict and too lenient, creating holes in one part of a scene and floating voxels elsewhere. The reconstructions in Figure 13 all demonstrate this to varying degrees.

In most space carving implementations there has been an implicit assumption that the pixel resolution is greater than the voxel resolution—that is, a voxel projects to a number of pixels in at least some of the images. We believe this is reasonable and expect the trend to continue because: 1) runtime grows faster with increasing voxel resolution than it does with increasing pixel resolution, and 2) the resolution of economical and readily available cameras keeps growing. We make use of this assumption in the adaptive standard deviation and histogram consistency tests. Steinbach et al. [45] have reported that they obtained better reconstructions when they precisely computed the projections of voxels into images, rather than using approximations, like splats. We have observed the same effect and therefore use scan conversion to determine voxel projections. We make the assumption in this section that the scenes being reconstructed are approximately Lambertian, and we use the RGB color space, except where noted.

3.1 Monotonic Consistency Tests

Kutulakos and Seitz assume monotonic consistency tests will be used with space carving. When such tests and full visibility are employed, space carving is guaranteed to yield the photo hull, the unique photo-consistent model that is a superset of all other photo-consistent models.

Seitz and Dyer [39] determine the consistency of a voxel V using the likelihood ratio test (LRT):

$$(n-1)s^2_{\pi^V} < \tau \tag{1}$$

where π^V is the set of pixels from which V is visible, s_{π^V} is the standard deviation of the colors of the pixels in π^V, n is the cardinality of π^V, and τ is a threshold that is determined experimentally. LRT has the virtue of being monotonic. However, because of the $(n-1)$ term in Equation 1, LRT has the disadvantage that voxels that are visible from more pixels are more likely to be carved. Nevertheless, as shown in Figure 13b, LRT can produce a reasonable reconstruction when photographs are available that sample the scene fairly uniformly.

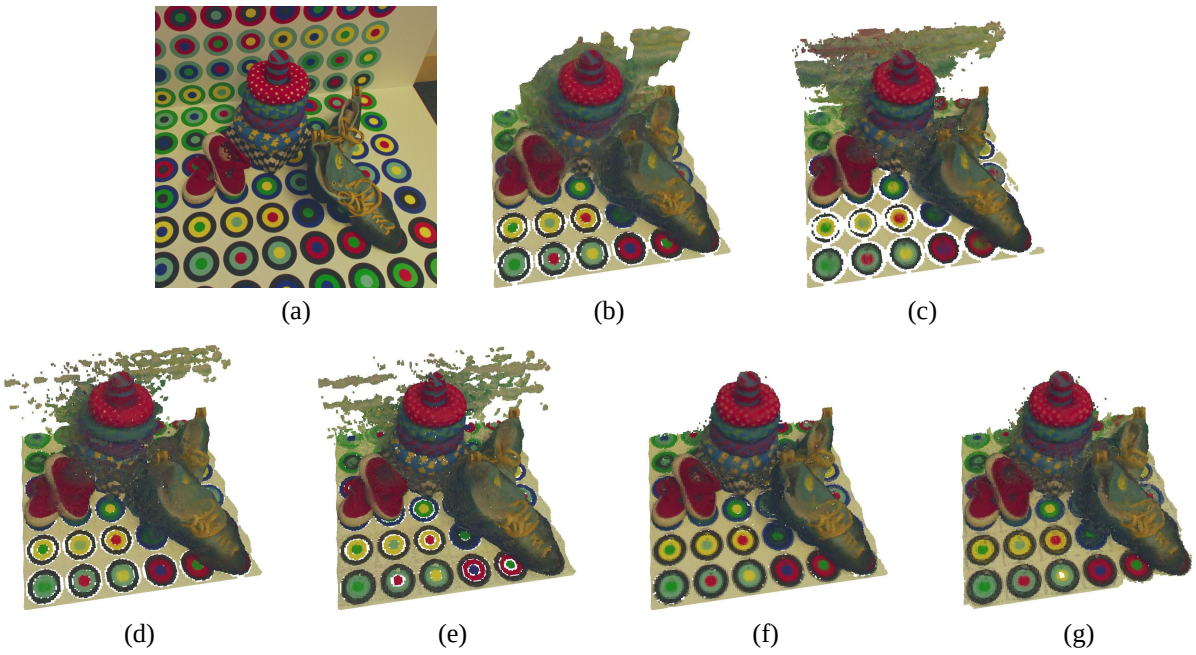


Figure 13: Reconstructions of the shoes data set using different consistency tests. (a) is a photograph of the scene that was not used during reconstruction. (b) was reconstructed using the likelihood ratio test, (c) using the bounding box test, (d) using standard deviation, (e) using standard deviation and the CIELab color space, (f) using the adaptive standard deviation test, and (g) using the histogram test.

The next two consistency tests we describe are monotonic and lack LRT's sensitivity to the number of pixels that view voxels. Perhaps the most obvious choice for a monotonic consistency test is:

$$max\{dist(color(p_1), color(p_2)) \mid p_1, p_2 \in \pi^V\} < \tau \quad (2)$$

where $dist$ is the L_1 or L_2 norm in color space. The disadvantages of this test are its computational complexity and its sensitivity to pixel noise. The bounding box test is a simple, related test with low computational complexity. In this test, a voxel V is checked for consistency by comparing a threshold to the length of the great diagonal of the axis-aligned bounding box, in RGB space, of the colors of the pixels in π^V. Disadvantages of the bounding box test are that it is a somewhat crude measure of color consistency and it is sensitive to pixel noise. A reconstruction performed with this test, shown in Figure 13c, produced more floating voxels than LRT but also recovered some detail that LRT missed.

3.2 Non-monotonic Consistency Tests

We can easily think of plausible consistency tests, for example tests that threshold common statistics like standard deviation:

$$s_{\pi^V} < \tau \quad (3)$$

Unfortunately, many such tests are not monotonic, including thresholded standard deviation. When space carving is used with non-monotonic consistency tests, it can carve a voxel that might be consistent in the final model. The algorithm can also converge to different models depending upon the order in which the voxels are processed, so there is no unique photo hull corresponding to such tests. However, this is not necessarily a disadvantage if the objective is to produce models that closely resemble the scene. In fact, among the tests we have tried, the two that have consistently produced the best looking models, the adaptive standard deviation test and the histogram test, are not monotonic. The reconstruction shown in Figure 13d, produced using thresholded standard deviation, demonstrates that non-monotonic tests can yield reasonable models. The test produced fewer floating voxels than LRT and the bounding box test, and recovered some detail that LRT missed.

3.2.1 An Adaptive Consistency Test

Because we consider voxels to have nonzero spatial extent, they can represent portions of surfaces that include abrupt color changes and significant texture. Using any of the consistency tests already described, a high threshold is

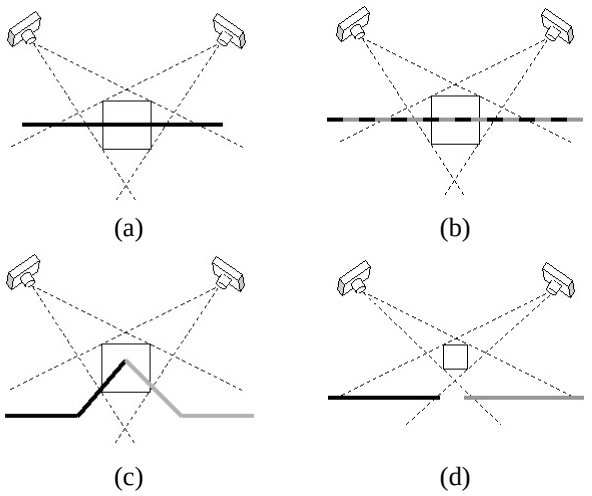


Figure 14: Handling texture and edges. In (a), a voxel represents a homogeneous region, for which both s_{π^V} and $\overline{s}$ are small. In (b) and (c), a voxel represents a textured region and an edge, respectively, for which both s_{π^V} and $\overline{s}$ are large. In (d), a voxel representing free space has a large s_{π^V} and small $\overline{s}$.

needed to reconstruct such surfaces. The same scenes can also include surfaces with little or no color variation. Such regions require a low threshold to minimize cusping and floating voxels. Fortunately, we can measure the amount of color variation on a surface by measuring the amount color variation the surface projects to in single images.

This suggests that it would be beneficial to use an adaptive threshold that is proportional to the color variation seen from single images. This is illustrated in Figure 14. Let π_i^V be the set of pixels in image i from which voxel V is visible, let $s_{\pi_i^V}$ be the standard deviation of π_i^V and let $\overline{s}$ be the average of $s_{\pi_i^V}$ for all images i from which V is visible. In (a), (b) and (c) in the figure, where the voxel is on the surface, note that s_{π^V} and $\overline{s}$ are both simultaneously either small or large. In (d), where the voxel is not on the surface, $\overline{s}$ is small and s_{π^V} is large.

We constructed an adaptive consistency test, which we call the adaptive standard deviation test (ASDT), as follows:

$$s_{\pi^V} < \tau_1 + \tau_2 \overline{s} \tag{4}$$

where τ_1 and τ_2 are thresholds whose values are determined experimentally. ASDT is the same as the thresholded standard deviation test of Equation 3 except for the $\tau_2 \overline{s}$ term.

Figure 15 shows a data set for which thresholded standard deviation, regardless of threshold, failed to reconstruct the scene, yet ASDT produced a reasonable model.

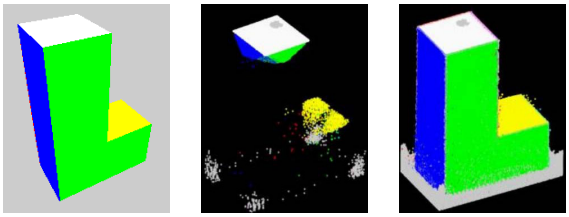


Figure 15: Figure (a) is a reference image from our El data set, (b) shows the best reconstruction obtained using thresholded standard deviation, and (c) shows a reconstruction obtained using the adaptive standard deviation test.

Figure 13f shows an ASDT reconstruction that is superior to reconstructions produced by any of the other consistency tests we have described so far. Note that the ASDT model has fewer floating voxels as well as fewer holes than the other models. A disadvantage of ASDT is the experimentation that is required to find the values of τ_1 and τ_2 that produce the best reconstruction.

3.2.2 A Histogram-Based Test

Since a voxel often represents a part of a surface that crosses color boundaries or includes significant texture, it can be visible from pixels whose colors have a complex, multi-modal distribution. A few parameters of a distribution, such as the variance and standard deviation used by the previous tests, can only account for second order statistics of the distribution. Furthermore, such parameters make assumptions about the distributions, for example standard deviation accurately characterizes only Gaussian distributions. The multi-modal color distributions that we would like to characterize are unlikely to conform to any such assumptions. In contrast, histograms are nonparametric representations that can accurately describe any distribution. This inspired us to develop a consistency test that uses histograms. The test produces excellent reconstructions and has the additional advantage of requiring little or no parameter adjustment.

Our histogram test computes a color histogram for each image from which a voxel is visible and then compares the histograms. There are many ways to compare histograms; we use histogram intersection because it produces good reconstructions and is simple and efficient to implement. We say two histograms intersect if there is at least one pair of corresponding bins (one bin from each histogram) in which both bins have nonzero count. For a given voxel V and image i, we build a histogram $Hist(\pi_i^V)$ of the colors of all the pixels in π_i^V. We define V to be consistent if, for every pair of images i and j for which π_i^V and π_j^V are not empty,

$Hist(\pi_i^V)$ and $Hist(\pi_j^V)$ intersect. In other words,

$$\forall_{i,j} Hist(\pi_i^V) \bigcap Hist(\pi_j^V) \neq \emptyset \quad i \neq j \qquad (5)$$

Therefore, a single pair of views can cause a voxel to be declared inconsistent if the colors they see at the voxel do not overlap. We use a 3D histogram over the complete color space. Furthermore, we have found that eight bins per channel are adequate for acceptable reconstructions; this yields 512 bins ($8 \times 8 \times 8$) for each image of each voxel.

We made several optimizations to minimize the runtime and memory requirements related to our consistency test. Notice that the histogram intersection only needs to test which histogram bins are occupied. Hence, only one bit is required per bin, or 512 bits per histogram. Histogram intersection can be tested with AND operations on computer words. Using 32-bit words, only 16 AND instructions are needed to intersect two histograms. The number of histogram comparisons needed to test the consistency of a voxel is equal to the square of the number of images that can see the voxel. Fortunately, in our data sets the average number of images that could see a voxel fell between two and three, so the number of histogram comparisons was manageable.

Histogram-based methods can suffer from quantization: a set of colors that falls in the middle of a histogram bin can be treated very differently from a set that is similar but is near a bin boundary. We avoided this problem by using overlapping bins, which, in effect, blur the bin boundaries. Specifically, we enlarged the bins to overlap adjacent bins by about 20 percent. A pixel with a color falling into multiple overlapping bins is counted in each such bin. This makes the consistency test insensitive to bin boundaries and small inaccuracies in color measurement.

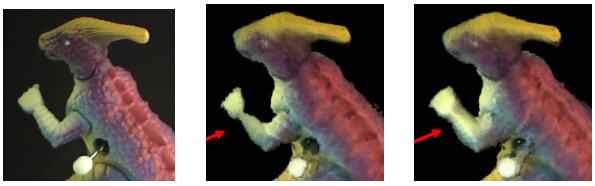
We found histogram intersections to be a somewhat unreliable indicator of color consistency when a voxel was visible from only a small number of pixels in some images. Hence, if this number fell below a fixed value (we typically used 15 pixels), we did not use the image in the consistency test.

Figure 16 shows a number of reconstructions produced with the histogram consistency test. The right column of the figure shows a reference view (that was used in the reconstruction), and the left image shows the reconstructed model projected to the same viewpoint as the reference view. Another histogram reconstruction, shown in Figure 13g, is similar to ASDT reconstruction but significantly better than the other reconstructions.

A significant advantage of our histogram consistency test is that it requires little or no parameter tuning. The test does have parameters, for example the number of histogram bins and the bin overlap, but the test is significantly less sensitive to its parameter settings than any of the other tests we have described. For example, using the thresholded standard deviation test and a typical data set, the threshold value that gave the best reconstruction was just 5% higher than a value that caused the reconstruction to fail catastrophically. With another data set but the same consistency test, the best reconstruction was obtained with a threshold 53% higher than the best value for the first data set. Finding ideal settings for sensitive parameters is very time consuming. In contrast, five of the six histogram reconstructions shown in Figure 16 were performed with our default parameters and no tuning was needed.

3.3 Color Spaces

The RGB color space, which we have used for most of our reconstructions, is not perceptually uniform. Specifically, in the RGB color cube, two dark colors that can be easily distinguished might be quite close together, whereas two bright colors that are relatively far apart might be hard to distinguish. It follows that, for a given threshold, a consistency test might be too lenient in dark areas, introducing floating voxels, while simultaneously being too strict in bright areas, introducing holes. There are many color spaces that would avoid this problem; we experimented with CIELab, which is perceptually uniform. Figures 17b and 17c show two reconstructions that are similar in most respects except a bright region in (c) was reconstructed more robustly in CIELab space than the corresponding region in (b), reconstructed in RGB. Figure 13e, also obtained using CIELab color space, is quite similar to the RGB reconstruction in Figure 13d, probably because the scene has relatively little brightness variation. The reconstructions in Figures 13e and 17 were produced using thresholded standard deviation, although CIELab should be equally effective with other consistency tests.



(a) (b) (c)

Figure 17: (a) is a photograph from the dinosaur data set, (b) was reconstructed in the RGB color space, and (c) shows a more robust reconstruction of a bright region obtained in the CIELab color space. Data set courtesy of Steve Seitz.

Because lighting can change while a scene is photographed, several people have suggested that we might obtain better results by weighting luminance less than

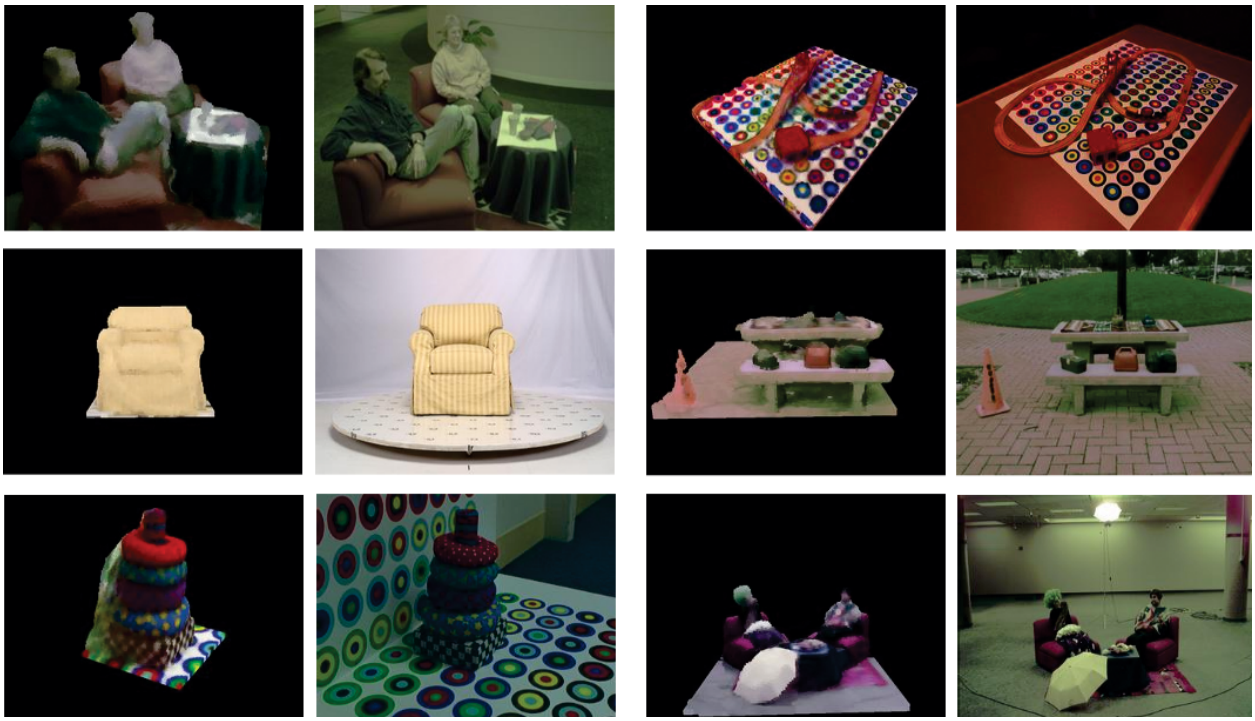


Figure 16: Reconstructions using the histogram intersection test for determining photo-consistency.

chrominance when testing color consistency. CIELab has a luminance channel, allowing us to test this idea. However, in several experiments, we found de-emphasizing luminance to be of minimal benefit.

3.4 Summary

Along with visibility, consistency tests have a large impact on the quality of reconstructions produced by space carving algorithms. We have described a number of consistency tests, including two we developed. Figure 13 allows the various consistency tests to be compared side-by-side. The adaptive standard deviation test and the histogram test yielded models that simultaneously have dramatically fewer floating voxels and holes. The histogram test avoids time-consuming experimentation because it is relatively insensitive to its parameter settings. We have also shown that the choice of color space can affect the quality of reconstructions, especially in unusually bright or dark regions of a scene.

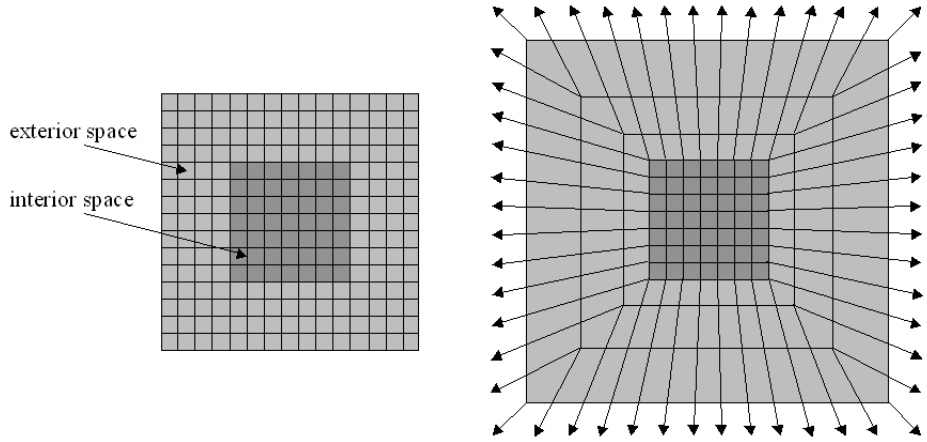
4 Volumetric Warping

We now present a volumetric warping technique for the reconstruction of scenes defined on an infinite domain. This approach enables the reconstruction of all surfaces, near to far away, as well as a background environment, using a voxel space composed of a finite number of voxels. By modeling such distant surfaces, in addition to foreground surfaces, this method yields a reconstruction that, when rendered, produces synthesized views that have improved photorealism.

Our volumetric warping approach is related to 2D environment mapping [3, 19] techniques that model infinite scenes for view synthesis. However, our approach is fully three-dimensional and accommodates surfaces that appear both in foreground and background. While methods that define the voxel space using the epipolar geometry between two or three basis views [37, 22] form a voxel space with variable resolution, our approach does not give preference to any particular reference views, and additionally, it extends the domain of the voxel space to infinity in all spatial dimensions.

4.1 Volumetric Warping Functions

The goal of a volumetric warping function is to represent an infinite volume with a finite number of voxels. The warping function must satisfy the requirements that in the warped space, no voxels overlap and no gaps exist between voxels. These requirements are easily accomplished for a variety of warping functions. We use the term *pre-warped* to refer to the volume before the warping function is applied.



(a) (b)

Figure 18: Pre-warped (a) and warped (b) voxel spaces shown in two dimensions. In (a), 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. Both regions consist of voxels of uniform size. The warped voxel space is shown in (b). 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 (b) are warped to infinity. These voxels are represented with arrows in the figure.

4.1.1 Frustum Warp

We now describe a frustum warp function that is used to warp the exterior space. We develop the equations and figures in two dimensions for simplicity; the idea easily extends to three dimensions.

The frustum warp function presented here separates the voxel space into an interior space used to model foreground surfaces at fixed resolution, and an exterior space used to model background surfaces at variable resolution, as shown in Figure 18. The warping function does not affect the voxels in the interior space, while voxels in the exterior space are warped so that their size increases linearly, in each dimension, with distance from the interior space. Under this construction, voxels in the exterior space will project to roughly the same number of pixels for viewpoints in or near the interior space. Voxels on the outer shell of the exterior space have coordinates warped to infinity, and have infinite volume. While the voxels in the warped space have a variable size, the voxel space still has a regular 3D lattice topology.

The frustum warp assumes that both the interior space and the pre-warped exterior space have rectangular shaped outer boundaries, as shown in Figure 19. These boundaries are used to define four trapezoidal regions, $\pm x$, and $\pm y$, based on the region's relative position to center of the interior space. These regions are also shown in Figure 19.

Let (x, y) be a pre-warped point in the exterior space, and let (x_w, y_w) be the point after warping. To warp (x, y),

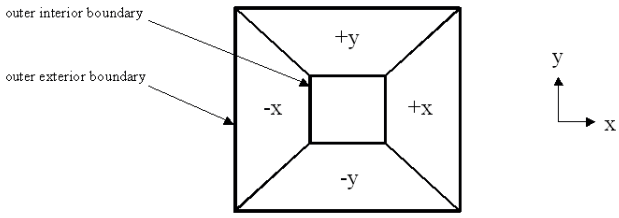


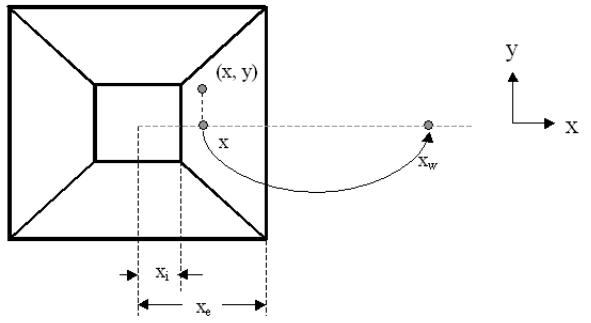
Figure 19: Boundaries and regions. The outer boundaries of both the interior and exterior space are shown in the figure. The four trapezoidal regions, $\pm x$ and $\pm y$ are also shown.

we first apply a warping function based on the region in which the point is located. This warping function is applied only to one coordinate of (x, y). For example, suppose that the point is located in the $+x$ region. Points in the $+x$ and $-x$ regions are warped using the x-warping function,

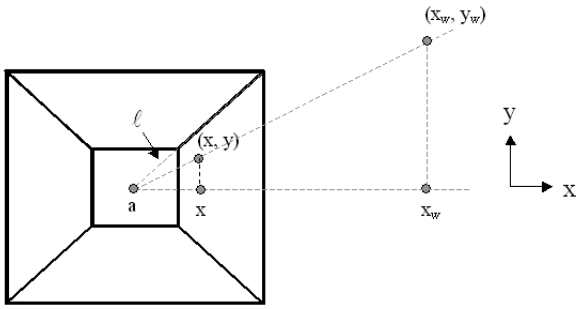
$$x_w = x \frac{x_e - x_i}{x_e - |x|},$$

where x_e is the distance along the x-axis from the center of the interior space to the outer boundary of the exterior space, and x_i is the distance along the x-axis from the center of the interior space to the outer boundary of the interior space, shown in (a) of Figure 20. A quick inspection of this warping equation reveals its behavior. For a point on the boundary of the interior space, $x = x_i$, and thus $x_w = x_i$,

so the point does not move. However, points outside of the boundary get warped according to their proximity to the boundary of the exterior space. For a point on the boundary of the exterior space, $x = x_e$, and so $x_w = \infty$.



(a)



(b)

Figure 20: Finding the warped point. The x-warping function is applied to the x-coordinate of the point (x, y), as the point is located in the $+x$ region. This yields the coordinate x_w, shown in (a). In (b), the other coordinate y_w is found by solving the line equation using the coordinate x_w found in (a).

Continuing with the above example, once x_w is computed, we find the other coordinate y_w by solving a line equation,

$$y_w = y + m(x_w - x),$$

where m is the slope of the line connecting the point (x, y) with the point a, shown in (b) of Figure 20. Point a is located at the intersection of the line parallel to the x-axis and running through the center of the interior space, with the nearest line l that connects a corner of the interior space with its corresponding matching corner of the exterior space, as shown in the figure. Note that in general, point a is not equal to the center of the interior space. By using such a construction, a point in a pre-warped region of space (e.g. $+x$) will stay in the that region after warping.

As shown above, the exterior space is divided into four trapezoidal regions for the two-dimensional case. In three dimensions, this generalizes to six frustum-shaped regions, $\pm x$, $\pm y$, $\pm z$; hence the term *frustum warp*. There are three warping functions, namely the x-warping function as given above, and y- and z-warping functions,

$$\begin{aligned} y_w &= y\frac{y_e - y_i}{y_e - |y|} \\ z_w &= z\frac{z_e - z_i}{z_e - |z|}. \end{aligned}$$

In general, the procedure to warp a point in the pre-warped exterior space is as follows.

- Determine in which frustum-shaped region the point is located.
- Apply the appropriate warping function to one of the coordinates. If the point is the in $\pm x$ region, apply the x-warping function, if the point is in the $\pm y$ region, apply the y-warping function, and if the point is the $\pm z$ region, apply the z-warping function.
- Find the other two coordinates by solving line equations using the warped coordinate.

4.1.2 Resolution in the Exterior Space

The pre-warped exterior space is warped to infinity in all directions by the warping function, regardless of how many voxels are in the exterior space, assuming that the exterior space consists of a shell of voxels at least one voxel thick in each direction. However, the number of voxels in the exterior space determines the resolution of the exterior space. Adding more voxels allows finer details of distant objects to be more finely modeled.

4.2 Implementation Issues

Reconstruction and new view synthesis of a scene using a warped voxel space poses some challenges, which we now describe.

First, the warped voxel space extends to infinity in each dimension, and therefore cameras get embedded inside the voxel space. Since the photo-consistency measure is effective only when each surface voxel is visible to two or more reference views, we must remove (pre-carve) a portion of the voxel space to produce a suitable initial surface for reconstruction. User guided or heuristic methods can be used for pre-carving.

Second, as discussed in Section 3, space carving algorithms sometimes carve voxels that should remain in the volume. Thus, it is possible that a voxel on the outer shell of the voxel space would be identified as inconsistent. Since one cannot see beyond infinity, we never carve voxels on the outer shell.

Finally, the geometry of objects in the exterior space is rather coarse since it is modeled with lower resolution voxels. After reconstruction, if a new view is synthesized near such low resolution geometry, the resulting image will appear distorted, as large individual voxels will be identifiable in the synthesized image. However, in our method, we intend for the reconstructed model to be viewed from in or near the interior space. For such viewpoints, objects in the exterior space will project to roughly a constant resolution that is matched to the outer shell of voxels in the interior space. This yields synthetic new views that do not suffer from such distortions.

4.3 Volumetric Warping Results

We have modified the GVC algorithm of Section 2 to utilize the warped voxel space. We performed a reconstruction using ten cylindrical panoramic photographs of a quadrangle at Stanford University. References [27, 41] discuss the calibration of such images. One of the 2500 x 884 photographs from the set is shown in Figure 21 (a). A voxel space of resolution 300 x 300 x 200 voxels, of which the inner 200 x 200 x 100 were interior voxels, was precarved manually by removing part of the voxel space containing the cameras. Then, the GVC algorithm was used to reconstruct the scene. A new synthesized view of the scene is shown in (b). Note that objects far away from the cameras, such as many of the buildings and trees, have been reconstructed with reasonable accuracy for new view synthesis.

Despite the successes of this reconstruction, it is not perfect. The sky is very far away from the cameras (for practical purposes, at infinity), and should therefore be represented with voxels on the outer shell of the voxel space. However, since the sky is nearly textureless, cusping [39] occurs, resulting in protrusions in the sky. Reconstruction of outdoor scenes is challenging, as surfaces often do not satisfy the Lambertian assumption made by our photo-consistency measure. On the whole, though, the reconstruction is accurate enought to produce convincing new views.

5 Conclusion

In this paper we have presented a collection of methods for the volumetric reconstruction of visual scenes. These methods have been developed to increase the quality, applicability, and usability of volumetric scene reconstruction. Visibility and photo-consistency are two essential aspects to any carving algorithm that reconstructs the photo hull. Accordingly, we introduced GVC, a full visibility reconstruction approach that supports arbitrary camera placement and does not process inner voxels. Our GVC-IB algorithm is memory efficient and easily hardware accelerated, while our GVC-LDI algorithm minimizes photo-consistency checks. We also presented new photo-consistency measures for use with Lambertian scenes. Our adaptive threshold method better reconstructs surface edges and texture, while our histogram intersection method requires nearly no parameter tuning. Finally, we showed how the voxel space can be warped so that infinitely large scenes can be reconstructed using a finite number of voxels.

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.

Acknowledgments

We thank Steve Seitz and Chuck Dyer for numerous discussions regarding volumetric scene reconstruction. We are grateful to Mark Livingston and Irwin Sobel for calibration of the Stanford data set. We also express our gratitude to Fred Kitson for his continued support and encouragement of this research.

References

[1] Beardsley, P., Torr, P., and Zisserman, A. 1996. 3D Model Acquisition from Extended Image Sequences. In *Proc. European Conference on Computer Vision*, pp. 683 - 695.

[2] Bhotika, R, Fleet, D., and Kutulakos, K. 2002. A Probabilistic Theory of Occupancy and Emptiness. In *Proc. European Conference on Computer Vision* Vol. 3, pp. 112 - 132.

[3] Blinn, J. and Newell, M. 1976. Texture and Reflection on Computer Generated Images. *Communications of ACM*, 19(10): 542 - 547.

[4] Bolles, R., Baker, H., and Marimont, D. 1987. Epipolar-Plane Image Analysis: An Approach to Determining Structure from Motion. In In *International Journal of Computer Vision* 1(1): pp. 7 - 55.

[5] Broadhurst, A. and Cipolla, R. 2001. A Probabilistic Framework for Space Carving. In *Proc. International Conference on Computer Vision*, pp. 388 - 393.

[6] Carceroni, R. and Kutulakos, K. 2001. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid 3D Motion, Shape & Reflectance. In *Proc. International Conference on Computer Vision*, pp. 60 - 67.

(a)

(b)

Figure 21: Stanford scene. One of the ten reference views, (a), and reconstructed model projected to a new synthesized panoramic view (b).

[7] Chhabra, V., 2001. Reconstructing Specular Objects with Image-Based Rendering Using Color Caching. Master's Thesis, Worchester Polytechnic Institute.

[8] Colosimo, A., Sarti, A., and Tubaro, S. 2001. Image-based Object Modeling: A Multi-Resolution Level-Set Approach. In *Proc. International Conference on Image Processing*, Vol. 2, pp. 181 - 184.

[9] Culbertson, W. B., Malzbender, T., and Slabaugh, G. 1999. Generalized Voxel Coloring. In *Proc. ICCV Workshop, Vision Algorithms Theory and Practice*, Springer-Verlag Lecture Notes in Computer Science 1883, pp. 100 - 115.

[10] De Bonet, J. and Viola, P. 1999. Roxels: Responsibility Weighted 3D Volume Reconstruction. In *Proc. International Conference on Computer Vision*, pp. 418 - 425.

[11] Debevec, P., Taylor, C., and Malik, J. 1996. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry and Image-Based Approach. In *Proc. SIGGRAPH*, pp. 11 - 20.

[12] Dyer, C. 2001. Volumetric Scene Reconstruction from Multiple Views. In Davis, L. S. ed., Foundations of Image Analysis. Kluwer: Boston, MA, pp. 469 - 489.

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

[14] Eisert, P., Steinbach, E., and Girod, B. 2000. Automatic Reconstruction of 3-D Stationary Objects from Multiple Uncalibrated Camera Views. In IEEE Transactions on Circuits and Systems for Video Technology, 10(2), pp. 261 - 277.

[15] Faugeras, O. and Lustman, F., 1988. Motion and structure from motion in a piecewise-planar environment. In *Journal of Pattern Recognition and Artificial Intelligence*, 2(3) pp. 485 - 508.

[16] Faugeras, O. and Keriven, R. 1998. Complete Dense Stereovision Using Level Set Methods. *IEEE Transactions on Image Processing*, 7(3): 336 - 344.

[17] Fua, P. and Leclerc, Y. 1995. Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 16(1): 35 - 56.

[18] Gortler, S., Grzeszczuk, R., Szeliski, R., and Cohen, M. 1996. The Lumigraph. In *Proc. SIGGRAPH*, pp. 43-54.

[19] Greene, N. 1986. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications*, pp. 21 - 29.

[20] Jebara, T., Azarbayejani, A., and Pentland, A. 1999. 3D structure from 2D motion. *IEEE Signal Processing Magazine*, 16(3): 66 - 84.

[21] Johansson, B. 1999. View Synthesis and 3D Reconstruction of Piecewise Planar Scenes Using Intersection Lines between the Planes. In *Proc. International Conference on Computer Vision*, 1999, Vol. 1, pp. 54 - 59.

[22] Kimura, M., Satio, H., and Kanade, T. 1999. 3D Voxel Construction Based on Epipolar Geometry. In *Proc. International Conference on Image Processing*, pp. 135 - 139.

[23] Kutulakos, K. and Seitz, S. 1999. A Theory of Shape by Space Carving. In *International Conference on Computer Vision*, Vol. 1, pp. 307 - 314.

[24] Kutulakos, K. and Seitz, S. 2000. A Theory of Shape by Space Carving. In *International Journal of Computer Vision*, 38(3), pp. 199 - 218.

[25] Laurentini, A. 1994. The Visual Hull Concept for Silhouette-Based Image Understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2): 150 - 162.

[26] Levoy, M. and Hanrahan, P. 1996. Light Field Rendering. In *Proc. SIGGRAPH*, pp. 31-42.

[27] Livingston, M. and Sobel, I. A New Scale Arbitration Algorithm for Image Sequences Applied to Cylindrical Photographs. HP Laboratories technical report HPL-2001-226R1, 2001.

[28] Longuet-Higgins, H., 1981. A Computer Algorithm for Reconstructing a Scene from Two Projections. *Nature*, 293(10): 133 - 135.

[29] Matusik, W., Buehler, C., Raskar, R., Gortler, S., and McMillan, L. 2000. Image-Based Visual Hulls. In *Proc. SIGGRAPH*, pp. 367 - 374.

[30] Max, N. 1996. Hierarchical Rendering of Trees from Precomputed Multi-Layer Z-Buffers. In *Proc. Eurographics Rendering Workshop*, pp. 165 - 174.

[31] McMillan, M. and Bishop, G. 1995. Plenoptic Modeling: An Image-Based Rendering System. In *Proc. SIGGRAPH*, pp. 39 - 46.

[32] Narayanan, P., Rander, P., and Kanade, T. 1998. Constructing Virtual Worlds Using Dense Stereo. In *Proc. International Conference on Computer Vision*.

[33] Özün, O. 2002. Comparison of Photo-Consistency Measures Used in the Voxel Coloring Algorithm. Masters thesis, Middle East Technical University.

[34] Pfister, H. Zwicker, M. Van Baar, J., and Gross, M. 2000. Surfels: Surface Elements as Rendering Primitives. In *Proc. SIGGRAPH*, pp. 335 - 342.

[35] Pollefeys, M., Koch, R., Vergauwen, M., and Van Gool, L. 1999. Hand-Held Acquisition of 3D Models With a Video Camera. In *Proc. 2nd International Conference on 3-D Digital Imaging and Modeling*, pp. 14 - 23.

[36] Prock, A. and Dyer, C. 1998. Towards Real-Time Voxel Coloring. In *Proc. DARPA Image Understanding Workshop*, pp. 315 - 321.

[37] Saito, H. and Kanade, T. 1999. Shape Reconstruction in Projective Grid Space from Large Number of Images. In *Proc. CVPR*, pp. 49 - 54.

[38] Savarese, S., Rushmeier, H., Bernardini, F., and Perona, P. 2001. Shadow Carving. In *Proc. International Conference on Computer Vision*, Vol 1, pp. 190 - 197.

[39] Seitz, S. and Dyer, C. 1999. Photorealistic Scene Reconstruction by Voxel Coloring. In *International Journal of Computer Vision*, 35(2): 151 - 173.

[40] Shade, J., Gortler, S., He, L., and Szeliski, R. 1998. Layered Depth Images. In *Proc. SIGGRAPH*, pp. 231-242.

[41] Shum, H. Y., Han, M., and Szeliski, R. 1998. Interactive Construction of 3D Models from Panoramic Mosaics In *Proc. CVPR*, pp. 427 - 433.

[42] Slabaugh, G., Malzbender, T., and Culbertson, W. B. 2000. Volumetric Warping for Voxel Coloring on an Infinite Domain. In *Proc. 3D Structure from Multiple Images for Large Scale Environments (SMILE)*, pp. 41 - 50.

[43] Slabaugh, G., Culbertson, W. B., Malzbender, T., and Schafer, R. 2001. A Survey of Volumetric Scene Reconstruction Methods from Photographs. In *Proc. International Workshop on Volume Graphics*, pp. 81 - 100.

[44] Slabaugh, G., Schafer, R., and Hans, M. 2002. Image-Based Photo Hulls. In *Proc. 1st International Symposium on 3D Processing, Visualization, and Transmission*, pp. 704 - 708.

[45] Steinbach, E., Eisert, P., Betz, A., and Girod, B. 2000. 3-D Reconstruction of Real World Objects Using Extended Voxels. In *Proc. International Conference on Image Processing*, vol. I, pp. 569 - 572.

[46] Stevens, M., Culbertson, W. B., and Malzbender, T. 2002. A Histogram-Based Color Consistency Test for Voxel Coloring. In *Proc. International Conference on Pattern Recognition*, vol. 4, p.118-21.

[47] Szeliski, R. 1993. Rapid Octree Construction from Image Sequences. *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(1): 23 - 32.

[48] Szeliski, R. and Golland, P. 1998. Stereo Matching with Transparency and Matting. In *Proc. International Conference on Computer Vision*, pp. 517 - 524.

[49] Tsai, R. 1987. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, RA-3(4): 323 - 344.

[50] Vedula, S., Baker, S., Seitz, S., and Kanade, T. 2000. Shape and Motion Carving in 6D. In *Proc. CVPR*, Vol. 2, pp. 592 - 598.

[51] Weghorst, H., Hooper, G., and Greenberg, D. P. 1984. Improving Computational Methods for Ray Tracing. *ACM Transactions on Graphics*, 3(1): 52 - 69.

[52] Yezzi, A. and Soatto, S., 2001. Stereoscopic Segmentation. In *Proc. International Conference on Computer Vision*, pp. 59 - 66.