

# Image-Based Photo Hulls for Fast and Photo-Realistic New View Synthesis

GREGORY G. SLABAUGH

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

RONALD W. SCHAFER

*Center for Signal and Image Processing, Georgia Institute of Technology, Atlanta, GA 30338*  
rws@ece.gatech.edu

MAT C. HANS

*Mobile and Media Systems Lab, Hewlett-Packard Laboratories, Palo Alto, CA 94304*  
mat\_hans@hp.com

## Abstract

*We present an efficient image-based rendering algorithm that generates views of a scene's photo hull. The photo hull is the largest 3D shape that is photo-consistent with photographs taken of the scene from multiple viewpoints. Our algorithm, Image-Based Photo Hulls (IBPH), like the Image-Based Visual Hulls (IBVH) algorithm from Matusik et. al. on which it is based, takes advantage of epipolar geometry to efficiently reconstruct the geometry and visibility of a scene. Our IBPH algorithm differs from IBVH in that it utilizes the color information of the images to identify scene geometry. These additional color constraints result in more accurately reconstructed geometry, which often projects to better synthesized virtual views of the scene. We demonstrate our algorithm running in a realtime 3D telepresence application using video data acquired from multiple viewpoints.*

## Keywords

Photo hull, image-based rendering, 3D photography, new view synthesis, voxel coloring, space carving, color consistency, view-dependent scene reconstruction.

## 1 Introduction

The task of generating a photo-realistic 3D representation of a visual scene is an important and challenging problem. Debevec et. al. [1] demonstrated in their Campanile movie that it is possible, using a user-assisted 3D modelling program and a handful of photos of a college campus, to produce a digital model of the scene that when rendered, yields images of stunning photorealism from new viewpoints. Since this work, there has been much interest in producing results of similar quality using algorithms that

are automatic and work on scenes composed of surfaces of arbitrary geometry.

Recently, researchers have become interested in reconstructing time-varying scenes [2, 3, 4, 5, 6]. Most standard approaches to the 3D scene reconstruction problem such as multi-baseline stereo, structure from motion, and shape from shading were not designed for realtime performance and thus are too slow to process the images online. When working with multi-view video data, most techniques perform the 3D reconstruction offline after the images have been acquired. Once the reconstruction is complete, it is rendered in realtime.

A notable exception is the Image-Based Visual Hulls (IBVH) algorithm [7], developed by Matusik et. al. This algorithm is efficient enough to reconstruct and render new views of the scene in realtime. The key to this algorithm's efficiency is its use of epipolar geometry for computing the geometry and visibility of the scene. By taking advantage of epipolar relationships, all of the steps of the algorithm function in the image space of the photographs (also called *reference views*) taken of the scene.

While the IBVH algorithm is exceptionally efficient, the geometry it reconstructs is not very accurate. This is because the IBVH algorithm only reconstructs the visual hull of the scene. The visual hull is a conservative shape that contains the scene surfaces being reconstructed. When photographed by only a few cameras, the scene's visual hull is much larger than the true scene. Even if photographed by an infinite number of cameras, many objects with concavities cannot be modelled correctly by a visual hull. One can partially compensate for such geometric inaccuracies by view-dependent texture-mapping (VDTM), as done in the IBVH approach.

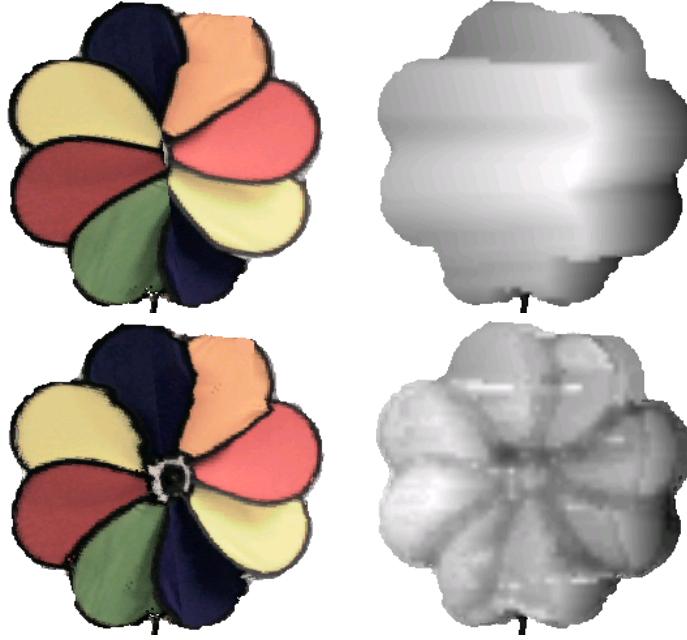


Figure 1: Visual hull reconstruction (upper row) vs. photo hull reconstruction (lower row) of a pinwheel. Left to right: synthesized view and depth map.

However, artifacts resulting from the inaccurate geometry are still apparent in new synthesized views of the scene, as shown in Figure 1. This figure demonstrates a reconstruction of a pinwheel photographed from five viewpoints. A new view of the scene, placed half-way between two reference views, is rendered from the reconstruction. The top row shows the visual hull reconstruction. At this viewpoint, the right side of the reconstructed pinwheel is texture-mapped with one reference image, while the left side of the pinwheel is texture-mapped with another. Due to the geometric inaccuracy of the visual hull, there is a salient seam along the pinwheel where there is a transition between the two images used to texture-map the surface. In particular, the center of the pinwheel is not present in the synthetic view. The improved geometry of the photo hull corrects this problem, as shown in the bottom row of the figure.

In this paper we adapt the IBVH algorithm to reconstruct photo hulls. The photo hull is the largest shape that is consistent with the photographs taken of the scene. Typically, the photo hull is also a shape that contains the scene surfaces; however it is a tighter fit to the true scene geometry than the visual hull. New views synthesized using the more accurate geometry of the photo hull have improved photorealism. Like IBVH, all the steps of our algorithm function in image space (hence, we call our algorithm *Image-Based Photo Hulls*). Our approach combines

the efficiency of the IBVH algorithm with the improved geometric accuracy of the photo hull.

The rest of this paper is organized as follows. In Section 2 we discuss related work. Since our algorithm extends the IBVH algorithm, we briefly review IBVH in Section 3. We describe the details of our approach in Section 4, and then present experimental results in Section 5 and evaluation in Section 6. Note that this paper is an expanded version of a symposium paper [9].

## 2 Related Work

### 2.1 Visual Hulls

A standard approach to reconstructing a 3D object using multi-view images is to compute the visual hull [8]. For each reference view, a silhouette is generated by segmenting the photograph into foreground and background. Foreground pixels correspond to points to which the 3D object projects. Everything else is background.

Each silhouette constrains the 3D space in which the object is located. If a 3D point projects to background in any of the images, it cannot be part of the 3D object being reconstructed. After eliminating such points, the surface of the region of space that remains is the *visual hull*. The visual hull is guaranteed to contain the 3D object. Using more reference views produces a visual hull that more closely resembles the geometric shape of the true 3D object. However, even with an infinite number of pho-

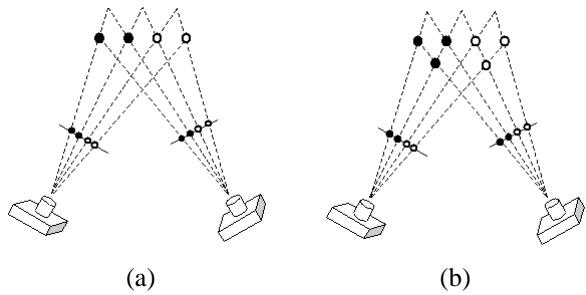


Figure 2: The problem of 3D scene reconstruction from multiple photographs is ill-posed.

tographs, the visual hull cannot model surface concavities that are not apparent in the silhouettes.

A variety of algorithms have been developed to compute the visual hull of a scene. Perhaps the most common approach is to operate in a volumetric framework. A good example is given in [10]. A volume that contains the scene being reconstructed is defined. The volume is then tessellated into voxels. All the voxels that project to solely background pixels in one or more reference views are removed (carved). The remaining voxels represent the visual hull and its interior. Such an algorithm can adopt a multi-resolution strategy to achieve faster results.

Recently, the Image-Based Visual Hulls [7] algorithm was developed and demonstrated to produce realtime new views of a scene. Our IBPH algorithm is based on the IBVH algorithm, so we will briefly review IBVH in Section 3.

## 2.2 Photo Hulls

The problem of reconstructing a 3D model of a scene using multiple 2D photographs is ill-posed. For a given set of 2D photographs, multiple 3D models that reproduce the photographs can and often do exist. For example, consider Figure 2. Here, two (of several) different models that reproduce the photographs are shown. Without more constraints on the reconstruction, it is impossible to determine which is a better representation of the true scene.

Given this ambiguity, Kutulakos and Seitz [11] introduce the *photo hull*, which is the *largest* shape that contains all reconstructions in the equivalence class of 3D models that reproduce the photographs. The photo hull can be thought of as the spatial union of all 3D models that reproduce the photographs. Since the true scene is one such 3D model, the photo hull is typically larger than the true scene. The photo hull is interesting because:

1. It is itself a photo-consistent reconstruction of the scene.

2. It is unique.
3. It can be easily computed.

A number of algorithms that compute photo hulls have been developed [11, 12, 13, 14]. These methods utilize photo-consistency [11] as a constraint to identify scene surfaces. A point in space is said to be *photo-consistent* if (1) it does not project to background and (2) when visible, the light exiting the point (i.e. radiance) in the direction of each reference view is equal to the observed color in the photograph.

For simplicity, one often assumes that the scene is Lambertian, although this isn't strictly necessary. Under this assumption, a point on a scene surface will project to a similar color in each reference view. The photo hull is then computed by finding the spatially largest set of points in 3D space that project to matching colors in the reference images.

These algorithms begin with a voxel space of initially opaque voxels that encompasses the scene to be reconstructed. As the algorithms run, opaque voxels are tested for photo-consistency and those that are found to be inconsistent are carved, i.e. made transparent. Convergence occurs when all the remaining opaque voxels are photo-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. This model is the photo hull. New, photo-realistic views of the scene can be synthesized by rendering the photo hull to virtual viewpoints.

Our IBPH algorithm adopts a similar strategy to compute views of the photo hull. The algorithm starts with a surface larger than the scene, and then iteratively carves space using a photo-consistency measure until the visible reconstructed points become photo-consistent. Our approach differs from previous photo hull reconstruction algorithms in that the IBPH algorithm functions in image space and produces a view-dependent reconstruction. Unlike [11, 12, 13, 14, 15], our approach does not employ a static 3D voxel space for reconstruction. Rather, the reconstructed geometry is computed in a space defined by a virtual camera, and changes as the camera is moved about the scene. Rather than reconstruct the full photo hull geometry, our method only reconstructs the portion of the photo hull that is visible to the virtual camera.

For an overview of methods that reconstruct visual and photo hulls, please refer to [16, 17].

## Notation

Before proceeding to the description of our algorithm, let us define some notation. A point in 3D space is represented in homogeneous coordinates by a boldface capital letter, such as  $\mathbf{P}$ . In this paper,  $\mathbf{P} = [x \ y \ z \ w]^T$ .

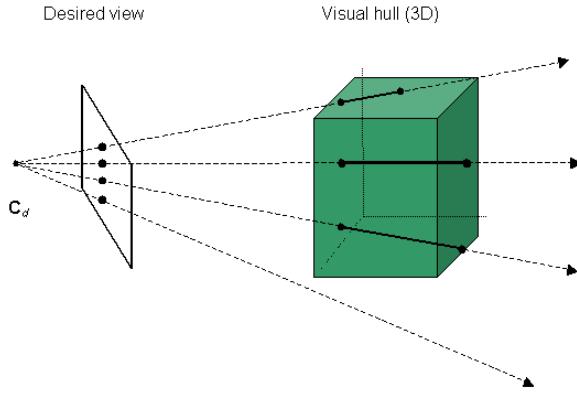


Figure 3: View-dependent geometry.

The projection of this point into an image is a 2D point represented in homogeneous coordinates by a boldface lowercase letter, such as  $\mathbf{p} = [x \ y \ w]^T$ . To convert a homogeneous image point to inhomogeneous coordinates (i.e. pixel coordinates), one simply divides  $\mathbf{p}$  by the  $w$  component. Thus, a pixel will have coordinates  $\mathbf{p} = [x/w \ y/w \ 1]^T$ .

### 3 Image-Based Visual Hulls

In this section, we briefly review the IBVH algorithm. In the following section, we will show how we extend the IBVH algorithm to reconstruct views of the photo hull.

#### 3.1 Computing geometry

One of the unique properties of the IBVH algorithm is that the geometry it reconstructs is view-dependent. A user moves a virtual camera about the scene. For each virtual camera placement (also called *desired view*), the IBVH algorithm computes the extent that back-projected rays from the center of projection  $\mathbf{C}_d$  intersect the visual hull in 3D space, stored as a layered depth image [18]. This is shown in Figure 3. Thus, the representation of the geometry is specified for the desired view, and changes as the user moves the virtual camera.

Consider an individual ray, as shown in Figure 4. The ray is back-projected from the desired view's center of projection, through a pixel in the image plane, and into 3D space. This ray projects to an epipolar line in each reference view. The IBVH algorithm determines the 2D intervals where the epipolar line crosses the silhouette. These 2D intervals are then “lifted” back onto the 3D ray using a simple projective transformation. The intervals along the 3D ray from all reference views are intersected. The resultant set of intervals describe where the ray pierces the visual hull. These are called *visual hull intervals* in this paper. In Figure 4, one visual hull interval is found along

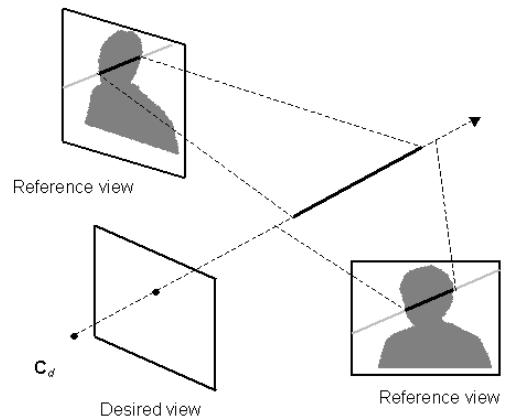


Figure 4: Determining a ray's visual hull intervals.

the back-projected ray. Once this procedure has been performed on all rays back-projected from the desired view, the reconstruction of the view-dependent geometry of the visual hull is complete.

#### 3.2 Computing visibility

In order to color a point on the visual hull, it is necessary to determine which cameras have an unoccluded view of the point. Thus, visibility must be computed before texture-mapping the reconstructed geometry.

At a pixel  $\mathbf{p}$  in the desired view, the first point (if any) along the first visual hull interval indicates a point  $\mathbf{P}$  in 3D space that projects to  $\mathbf{p}$  and is visible in the desired view, as shown in Figure 5 (a). To compute visibility, for each reference view we need to determine if  $\mathbf{P}$  is visible.  $\mathbf{P}$  must be visible in the reference view if the line segment  $\overline{\mathbf{P}\mathbf{C}_r}$  between  $\mathbf{P}$  and the reference view's center of projection  $\mathbf{C}_r$  does not intersect any visual hull geometry.

The layered depth image representation of the visual hull makes this easy to determine. In the desired view,  $\overline{\mathbf{P}\mathbf{C}_r}$  projects to an epipolar line segment  $\overline{\mathbf{p}\mathbf{e}}$ , where  $\mathbf{e}$  is the epipole, found by projecting  $\mathbf{C}_r$  into the desired view, as shown in Figure 5 (b). For each pixel along  $\overline{\mathbf{p}\mathbf{e}}$ , the visual hull intervals can be checked to see if they contain geometry that intersects  $\overline{\mathbf{P}\mathbf{C}_r}$ . If an intersection occurs, point  $\mathbf{P}$  is not visible in the reference view, and no more pixels along  $\overline{\mathbf{p}\mathbf{e}}$  need be evaluated. Otherwise, one continues evaluating pixels along  $\overline{\mathbf{p}\mathbf{e}}$ , until there are no more pixels to evaluate. If no visual hull interval has intersected  $\overline{\mathbf{P}\mathbf{C}_r}$ , then the point  $\mathbf{P}$  is visible in the reference view.

The IBVH paper [7] discusses discretization issues in computing visibility using this approach, as well as occlusion-compatible orderings to improve its efficiency.

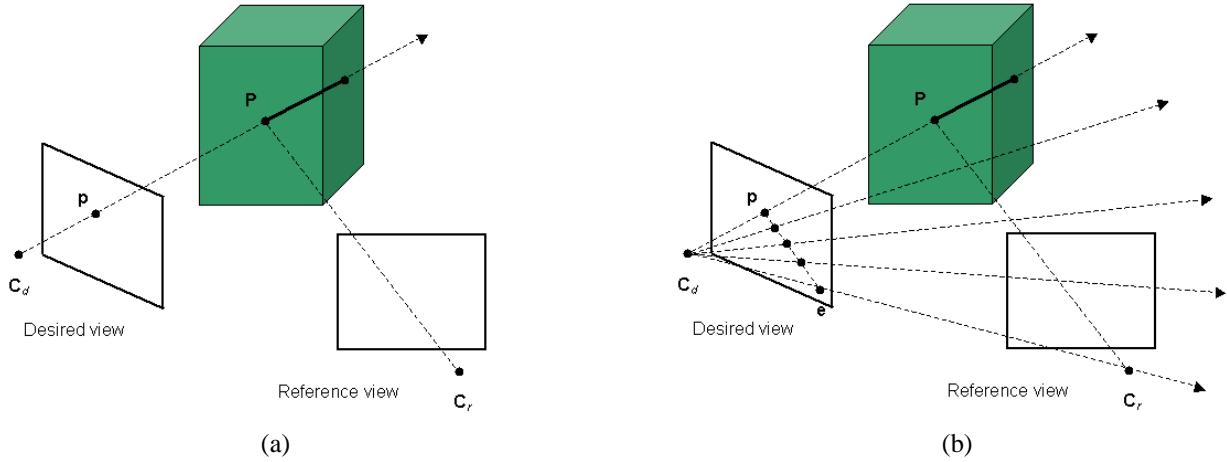


Figure 5:  $\mathbf{P}$  is visible in the reference view if there is no occluding geometry along  $\overline{\mathbf{P}C_r}$ .

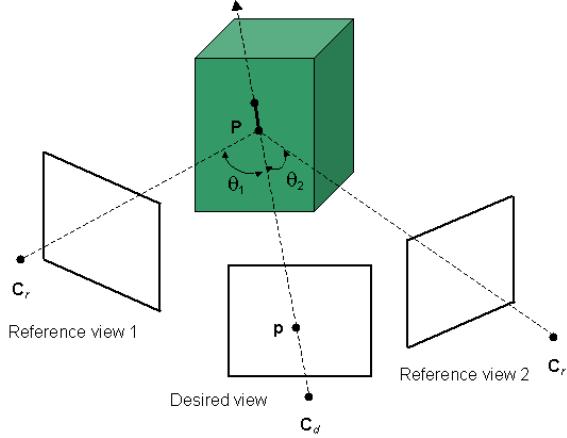


Figure 6: View-dependent texture-mapping.

### 3.3 View-dependent texture mapping

Once visibility has been computed, one can color the visual hull using the reference views. The IBVH paper employs view-dependent texture mapping, which retains view-dependent effects present in the photos, and works well with the inaccurate geometry of the visual hull. To color a point  $\mathbf{p}$  in the desired view, the closest point  $\mathbf{P}$  on the hull is found. Then, for each reference view that has visibility of  $\mathbf{P}$ , the angle between  $\overline{\mathbf{P}C_d}$  and  $\overline{\mathbf{P}C_r}$  is found, as shown in Figure 6. The reference view with the smallest angle is chosen to color the visual hull. This is the reference view that has the “best” view of  $\mathbf{P}$  for the virtual camera’s location. For example, in Figure 6, reference view 2 would be chosen since  $\theta_1 > \theta_2$ .

## 4 Image-Based Photo Hulls

In this section we describe our IBPH algorithm, which produces views of the photo hull.

### 4.1 Approach

Our IBPH approach first computes the visual hull using the IBVH algorithm, which quickly eliminates a large portion of 3D space that does not contain scene surfaces. Our algorithm then evaluates the photo-consistency of the closest point on the visual hull along each ray back-projected from the desired view. If the point is inconsistent, we take a small step along the ray, moving away from the desired view, as depicted in Figure 7. We continue stepping along an inconsistent ray until it either becomes consistent or we have stepped beyond all visual hull intervals along the ray. This latter case indicates that no photo-consistent geometry along the ray was found.

Note that in this approach, only the points on the hull that are visible in the desired view are processed. Initially, these points are the first points in the first visual hull interval along each back-projected ray. By stepping along the inconsistent rays until convergence, the IBPH algorithm reconstructs only the portion of the photo hull that is visible to the desired view.

### 4.2 Photo-Consistency

To determine the photo-consistency of a 3D point  $\mathbf{P}$  along a ray, we project  $\mathbf{P}$  into the  $i$ th reference view, yielding an image-space point  $\mathbf{p}_i$ . We only perform this projection for the reference views that have visibility of  $\mathbf{P}$ . Around each  $\mathbf{p}_i$  we collect a small neighborhood of pixels,  $N_i$  to use in our color matching function.

There are many methods one can employ for matching color distributions to determine photo-consistency. A stan-

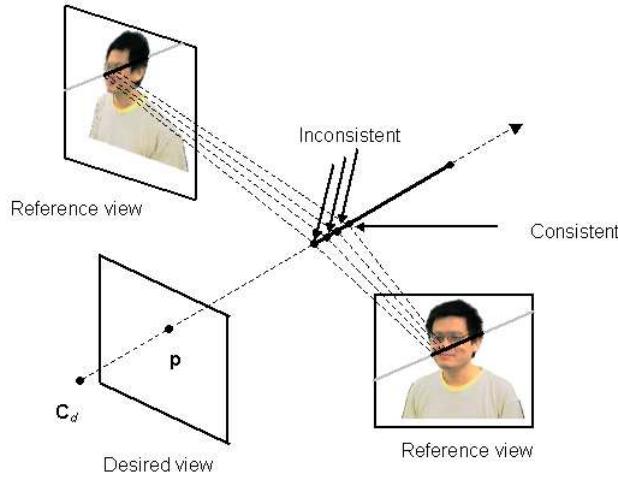


Figure 7: Computing the image-based photo hull.

standard approach is to threshold the standard deviation of all the pixels. That is,

$$\text{consistency} = \begin{cases} \text{True, if } \sigma \leq T_1 \\ \text{False, otherwise} \end{cases} \quad (1)$$

where  $\sigma$  is the standard deviation of all pixels,  $\sum_i N_i$ , and  $T_1$  is a user-defined threshold. In our work, we reconstruct the scene using RGB images, so we compute the standard deviation as  $\sigma = \sqrt{\sigma_r^2 + \sigma_g^2 + \sigma_b^2}$ , where  $\sigma_r$ ,  $\sigma_g$ , and  $\sigma_b$  are the standard deviations computed in the red, green, and blue color channels, respectively.

This measure of photo-consistency works reasonably well for many scenes. However, it can perform poorly for surfaces that project to consistent, yet highly varying colors in the reference images. This can occur on edges as well as textured surfaces. In such a case, each reference image will observe multiple different colors for the surface. Consequently, the standard deviation will be high, and the photo-consistency measure can incorrectly return false.

Textured surfaces and edges will project to pixels with a high standard deviation in *each* image. We use this fact to modify the consistency measure described above to handle such surfaces. Let the standard deviation of a set of pixels  $N_i$  from a reference view be  $\sigma_i$ . Our new photo-consistency measure is then

$$\text{consistency} = \begin{cases} \text{True, if } \sigma \leq T_1 + \bar{\sigma}T_2 \\ \text{False, otherwise} \end{cases} \quad (2)$$

where  $\bar{\sigma}$  is the average of  $\sigma_i$  and  $T_2$  is a second user-defined threshold.

This consistency measure simply adds an additional term  $\bar{\sigma}T_2$  to the one defined in Equation 1. This term spatially adapts the consistency measure based on the colors

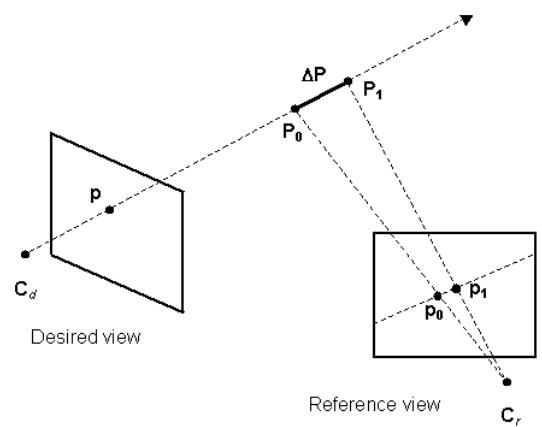


Figure 8: Stepping along an epipolar line.

observed in the 3D point's projection. The value of  $\bar{\sigma}$  will be small when the 3D point projects to homogenous colors in each image. In this case, there will be little difference between the two consistency measures 1 and 2. If these colors are similar, the point will be declared consistent. If these colors are dissimilar, the point will be declared inconsistent. When the point projects to highly varying pixels in each image, the  $\bar{\sigma}$  term will increase the maximum value of  $\sigma$  allowable for the point to be declared consistent. This allows for textured surfaces, as well as edges, to be correctly reconstructed. It also eases the Lambertian assumption. The threshold  $T_2$  allows one to weight the contribution of this adaptive term to the photo-consistency measure. This two-parameter consistency measure is the one used to produce photo hulls in this paper.

### 4.3 Stepping Along Epipolar Lines

As we step in 3D along an inconsistent ray, looking for the point at which it becomes consistent, we must simultaneously step along an epipolar line in each reference view. The brute-force way of stepping along the epipolar line in a reference view is to simply project each 3D point  $P_i$  on the ray to the reference view point  $p_i$  by multiplying the reference view's projection matrix  $H$  with  $P_i$ , i.e.  $p_i = HP_i$ . Such an approach will work, but will require a large number of matrix multiplications.

While the step size  $|\Delta P|$  in 3D is constant, the step size between adjacent points along the epipolar line in a 2D reference view varies due to the projection. However, since the projection is a homography (linear projective transformation), the step size is constant in homogeneous coordinates. We use this fact to produce a more efficient procedure for stepping along the epipolar line.

Consider the 3D point  $P_0$  on the ray, as shown in Figure 8. It projects to a point  $p_0 = HP_0$  in a reference

image. If we take a step along the ray, we arrive at a 3D point  $\mathbf{P}_1 = \mathbf{P}_0 + \Delta\mathbf{P}$ . The point  $\mathbf{p}_1$ , the projection of  $\mathbf{P}_1$  into the reference view can be written as

$$\begin{aligned}\mathbf{p}_1 &= H\mathbf{P}_1 \\ &= H(\mathbf{P}_0 + \Delta\mathbf{P}) \\ &= \mathbf{p}_0 + H\Delta\mathbf{P}\end{aligned}$$

Thus, we can incrementally update the homogeneous position of the point along the epipolar line in the reference view. That is,

$$\mathbf{p}_i = \begin{cases} H\mathbf{P}_0, & i = 0 \\ \mathbf{p}_{i-1} + H\Delta\mathbf{P}, & i > 0 \end{cases} \quad (3)$$

We set  $|\Delta\mathbf{P}|$  to a size that results in a projected size  $|\Delta\mathbf{p}|$  of roughly one pixel for most reference views. We then pre-compute the constant  $H\Delta\mathbf{P}$  for each ray in a reference view and store it in a look-up table. As we step along the epipolar line, we use Equation 3 to compute the homogeneous position of the point  $\mathbf{p}_i$ . With this approach, stepping along an epipolar line is very efficient.

#### 4.4 IBPH Visibility

When evaluating the photo-consistency of a 3D point, only pixels from the reference views that have visibility of the 3D point should be used. As one steps along the inconsistent rays, the visibility of the scene may change. A point that was not visible in a reference view before may become visible after the step is taken. Therefore, it is necessary to update visibility after stepping. This is achieved by re-executing the visibility procedure described in Section 3.2.

Visibility could be updated each time a step is taken along each ray. However, such an excessive number of visibility updates results in a slow reconstruction. Instead, our algorithm takes one step along each inconsistent ray, and then updates visibility. As a result, the visibility may be out-of-date when evaluating some 3D points. However, such an approach is conservative. Pixels from only a subset of the references views that have visibility of the point will contribute to the consistency measure. For a monotonic photo-consistency function [11], this may result in some 3D points being erroneously classified as consistent, while a full visibility calculation would show that they are really inconsistent. Since visibility is updated periodically, such erroneous classifications are properly classified on a later iteration of the algorithm. Such an approach is similar to that used in the GVC-IB [13] algorithm.

#### 4.5 Sampling

One way to trade off accuracy for speed in both the IBVH and IBPH algorithms is to compute the hull in a

multi-resolution fashion<sup>1</sup>. The algorithm is executed not for every pixel in the desired view, but rather on a coarse raster. One first computes the hull at sampling locations  $(x \cdot \text{DX}, y \cdot \text{DY})$  in the desired image, where  $\text{DX}$  and  $\text{DY}$  are constants that specify the sampling size. The sampling locations are shown as black dots in Figure 9. For in-between pixels on the boundary, indicated using black squares in the figure, the hull is computed at every pixel so that the edges of the synthesized image are at full resolution. For pixels inside the boundary, the closest point of the hull interval (i.e. depth) is interpolated from adjacent samples. This approach significantly reduces the number of rays that must be processed, resulting in a faster reconstruction.

Figure 10 shows the effect increasing the sampling size  $\text{DX}$  and  $\text{DY}$  for a pinwheel photographed from five viewpoints. The leftmost image shows the depth map when  $\text{DX}$  and  $\text{DY}$  are 1. In this case, we are computing a depth value for every pixel in the desired image. While the depth map is crisp, the frame rate is only 0.4 frames per second (FPS). Increasing the sampling size has a significant impact on the frame rate. For the rightmost image, the  $\text{DX}$  and  $\text{DY}$  are both five, and the frame rate is 8.3 FPS. The tradeoff for this improvement in frame rate is blurring of the depth map, since the depth values at pixels inside the border are interpolated from the sampling locations. Continuing to increase the sampling size further blurs the depth map, but has little impact on the frame rate, as more pixels become boundary pixels, which are sampled at full resolution.

#### 4.6 Convergence

The IBPH algorithm steps along the inconsistent rays, stopping at the point at which each ray becomes photo-consistent. For convergence, one can require that all rays are photo-consistent. However, often during a reconstruction, a significant majority of the rays will become consistent quickly. Continuing to process a handful of inconsistent rays will yield little impact on the overall quality of the reconstruction, but can take a lot of time. In our implementation, we have introduced a mechanism to terminate the reconstruction when  $M$  or less rays are inconsistent. When  $M$  is a small number, good quality hulls are produced quickly.

Figure 11 justifies our use of  $M$  to terminate the reconstruction before all rays are consistent. This plot shows ray classifications versus iteration for reconstruction of our Sam data set, which consists of a person's head photographed from four viewpoints. The visual hull projected to 1333 of the 80 x 60 points on the coarse raster. Rays back-projected through these points were analyzed using the IBPH algorithm. Initially, 635 were inconsistent and 698 were consistent, as shown in the figure. At each iter-

<sup>1</sup><http://graphics.lcs.mit.edu/~wojciech/vh/ctof.html>

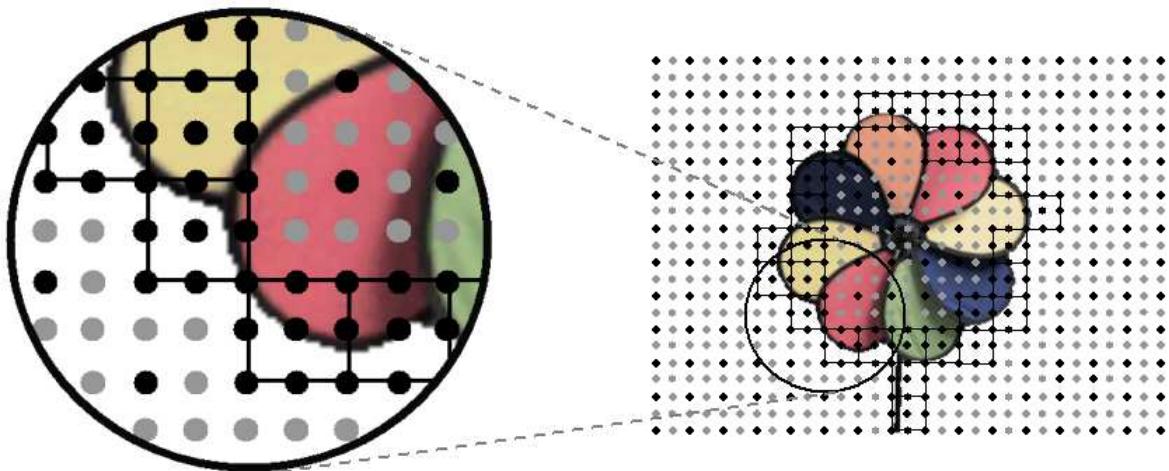


Figure 9: Multi-resolution sampling for faster performance. In this example,  $DX = DY = 2$ .

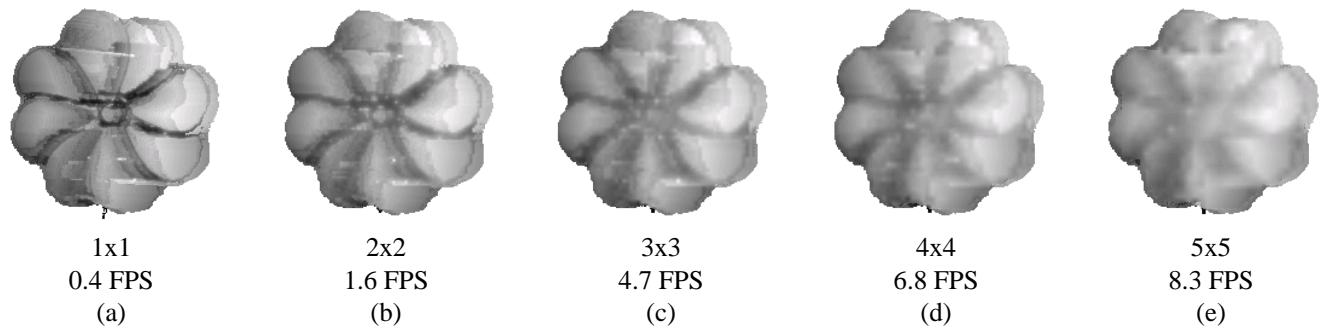


Figure 10: Effect of increasing sampling size  $DX$ ,  $DY$  on the reconstructed depth map and frame rate.

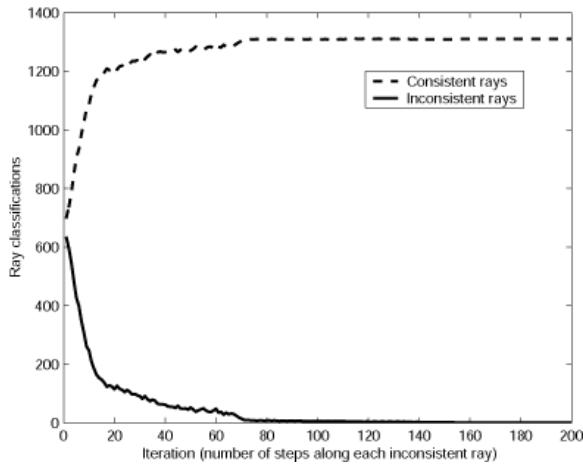


Figure 11: Ray classifications vs. iteration.

ation of the algorithm, a step was taken along each inconsistent ray. The plot of the number of inconsistent rays is very steep at first, indicating that many rays become consistent quickly. After 60 iterations, most rays are consistent. However, it takes an additional 140 iterations for the few remaining inconsistent rays to become consistent. For a realtime application, one would rather not continue processing these rays, as they will not significantly contribute to the quality of the reconstructed model.

There are several ways one might specify the value of  $M$ . For example, one may chose to set  $M$  to be a certain percentage of the total number of rays on the coarse raster. Alternatively, the value can be determined through experimentation. One can run a experiment like that done to produce Figure 11 to determine at what point the reconstruction is mostly complete for views of a scene. This is the technique used to produce the reconstructions in this paper. In another approach, one could terminate the reconstruction once the absolute value of the slope of a curve in the plot in Figure 11 goes below a threshold. Effectively, such a method would adaptively set the value of  $M$ .

For scenes with deep concavities that must be carved out from the visual hull, one may choose a lower value of  $M$  to ensure enough carving occurs before the reconstruction terminates.

#### 4.7 Spatial Coherence

Most scenes exhibit a high degree of spatial coherence, as they consist of surfaces that do not radically change their position over a small region of space. Accordingly, many stereo vision algorithms impose a regularization criterion that requires the reconstructed geometry to be smooth. In a similar vein, we have developed a very simple and efficient smoothing technique that we incorporate into our IBPH al-



Figure 12: Varying  $K$ . From top to bottom,  $K = 1, 2$ , and  $5$ .

gorithm. The smoothing also helps mitigate reconstruction errors due to noise and specularities.

When stepping along an inconsistent ray, we keep track of the number of steps we have taken,  $k$ . Before taking another step, we compare  $k$  to a local mean computed by averaging the number of steps taken along rays in a small neighborhood around the inconsistent ray. We denote this local average  $\bar{k}$ . If  $k > \bar{k} + K$ , where  $K$  is a small constant, we do not step along the ray. This ensures that the number of steps taken along a ray is not significantly different from that of its neighbors, resulting in a surface that is spatially coherent. This smoothing approach requires very little computation and works naturally with the representation of the hull geometry used in our algorithm.

Figure 12 shows the effect of changing  $K$  for a reconstruction of a person's head. Notice that there are less abrupt transitions in the depth map in the top-most reconstruction ( $K = 1$ ) compared to the bottom-most reconstruction ( $K = 5$ ). We note that this very simple smoothing approach is not ideal for reconstructing surfaces with large depth discontinuities observed from a small region in the desired view. For such situations, the smooth-

ing mechanism could be turned off. To better handle such scenes, more sophisticated (and most likely, more compute-intensive) smoothing methods would be required to analyze the 3D position of surface points to ensure that depth discontinuities are properly preserved. In this paper, we do not reconstruct such scenes so we will apply the smoothing method to all experimental results presented in Section 5.

#### 4.8 Homogeneous Surfaces

Surfaces that have a homogeneous color are difficult to properly reconstruct using color matching methods. A point in space near the surface will project to similar colors in the reference images. Such a point will be photo-consistent even though it is not actually on the surface being reconstructed. This results in cusps [12], which are 3D protrusions in the direction of the cameras, a common artifact seen in photo hulls. Our IBPH algorithm is not immune to this problem, as is visible in any of the reconstructions shown in Figure 12. The depth map indicates some extra geometry jutting out of the person’s chest, resulting from a homogeneously colored shirt the person was wearing. Fortunately, geometrical inaccuracies due to homogeneous surfaces are not that significant for new view synthesis, once the model is texture-mapped. For example, the seams shown in Figure 1 will not be present because the geometry will project to a homogeneous color in each reference view.

#### 4.9 Pseudo-Code

The pseudo-code for our IBPH algorithm appears in Figure 13.

### 5 Results

We have implemented the IBPH algorithm on a multi-camera system. We have five calibrated Sony DFW-V500 digital cameras. The cameras are synchronized so that they take images of the scene at the same instant of time. Each camera is connected to an 800 MHz HP Kayak machine. These machines perform background subtraction on the incoming frames, segmenting them into foreground and background regions. The resolution of the reference images is 320 x 240 pixels.

The segmented reference images are sent over a 100 Mb/s switch to our server machine, which computes the 3D reconstruction. Our server machine is a dual processor 2 GHz HP x4000 workstation. Our algorithm has been multi-threaded to take advantage of our multi-processor machine. The  $i$ th thread reconstructs the scene using a set of images corresponding to time  $t_i$ . In this way, the IBPH algorithm can very naturally be parallelized.

The bottom row of Figure 1 shows the results of IBPH algorithm in reconstructing a pinwheel. We placed the pinwheel in front of the cameras and spun the wheel. Fig-

ure 1 shows the reconstruction at one time instant. For this reconstruction, the sampling rate parameters  $DX$  and  $DY$  were 4, and the resolution of the desired view was 320 x 240. The algorithm reconstructed the scene and generated new views at 6 FPS. The IBPH algorithm produces more geometrically accurate results than the IBVH algorithm. However, the IBVH algorithm ran at 25 FPS for this data.

Figure 14 shows a view from a realtime 3D telepresence application we are currently developing with HP labs. The 3D model of the person’s head and neck is reconstructed online using the IBPH algorithm. The reconstructed geometry of the person is then depth-composited with a 3D model of a conference room. New synthesized views of this composited scene are generated at 7.5 frames per second. The upper image in the figure shows the texture-mapped model, while the lower image shows the depth map.

### 6 Evaluation

When working with this class of algorithms, a problem that often arises is evaluation. How can one objectively quantify the quality of a 3D scene reconstruction used in a new view synthesis application?

An intuitive approach is to use 3D ground truth information. When there is known 3D geometry, one can compute a 3D spatial error measure that characterizes the mismatch between actual and reconstructed geometry. While such a 3D spatial error measure is often quite insightful, it can potentially be an inadequate indicator for how well a reconstructed scene will produce new views. An alternative evaluation approach measures the 2D error of new views synthesized from the reconstruction. If there are  $N$  reference views, the reconstruction is performed using  $N - 1$  of them. The reconstructed model is then projected to the reference view that was left out, forming an image. This projected image is then compared with the reference view that was left out.

To better characterize the IBPH algorithm, and to compare it to the IBVH and GVC algorithms, we perform both a 3D spatial error analysis and a 2D new view synthesis error analysis for a synthetic data set we call *SynthPlane*, which consists of views of a multi-colored plane on the  $xy$  axis. The plane was photographed from 24 different viewpoints, each with a radius  $r = 10.5$  units from the center of the plane. Figure 15 shows the camera placements, as well as a bounding box containing the plane. Viewpoints were placed at  $45^\circ$  intervals of azimuth angle  $\theta$  and elevation angles of  $\phi = 15^\circ, 40^\circ$ , and  $65^\circ$ . Figure 16 shows three of the reference views for  $\theta = 0^\circ$  and  $\phi = 15^\circ, 40^\circ$ , and  $65^\circ$ . These images have a resolution of 640 x 480 pixels.

While a multi-colored plane is not the most exciting surface to reconstruct, it is useful and interesting for several

```

compute IBVH
compute visibility
pre-compute homogeneous ray steps  $H\Delta\mathbf{P}$  in each reference image
do
    evaluate photo-consistency
    for each inconsistent ray in desired view
        if (number of steps along ray  $k \leq \bar{k} + K$ )
            step along inconsistent ray
        else
            set ray consistent
    if (updating visibility)
        update visibility
} while(number of inconsistent rays >  $M$ )
display hull using VDTM

```

Figure 13: Pseudo-code for the IBPH algorithm. See text for details.



Figure 14: Using IBPH in a realtime 3D telepresence application.

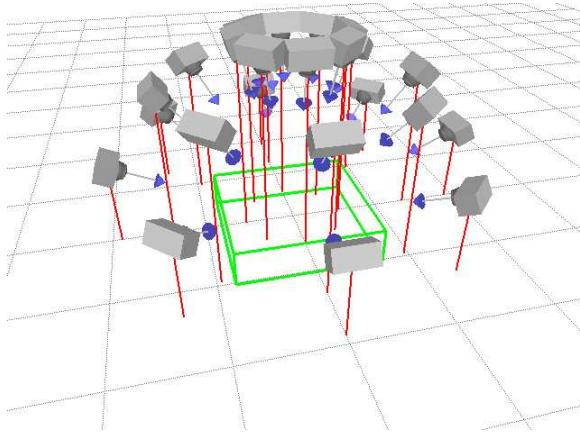


Figure 15: Camera placements and reconstruction volume for the SynthPlane scene.

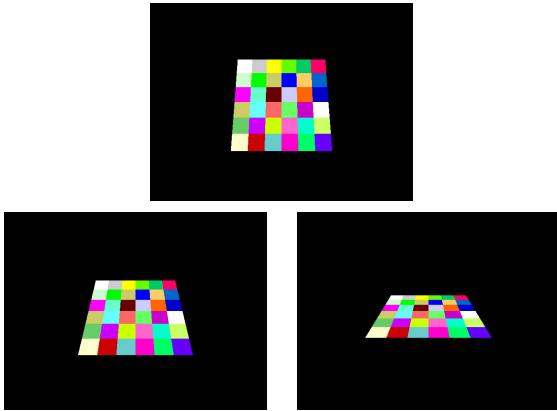


Figure 16: Three of 24 reference views for the SynthPlane scene, for  $\phi = 15^\circ$  (top),  $40^\circ$ , (bottom left) and  $65^\circ$  (bottom right).

reasons. First, it is easy to compare reconstructed geometry to that of a plane; we will describe our method below. Second, while the planar geometry is simple, it is effective for comparing different reconstruction algorithms since the reconstructed geometry exhibits the characteristics present in reconstructions of more geometrically complex surfaces.

## 6.1 3D Error

Our 3D error metric  $E_{3D}$  measures the volume of space that exists between the reconstructed surface and true surface located on the  $z = 0$  plane. This volume can be determined by solving the double integral,

$$E_{3D} = \int \int |h(x, y)| dx dy, \quad (4)$$

where  $h(x, y)$  is the height of reconstructed surface. We take the absolute value of  $h(x, y)$  to ensure that all contributions to the 3D error are positive. We call  $|h(x, y)|$  the *height field*.

For each reconstruction approach, we reconstruct the scene three times, using the highest 8 views, the highest 16 views, and all 24 views shown in Figure 15. Performing these three reconstructions shows the effect of reconstructing the scene using more oblique views. Since the IBVH and IBPH algorithms view-dependently reconstruct the scene, we placed the virtual camera directly above the  $xy$  plane, for an elevation angle  $\phi = 0^\circ$ , and a radius 10.5 units up the  $z$  axis. Using a sampling lattice of  $4 \times 4$ , we generated a synthetic view with resolution  $640 \times 480$  pixels. IBVH and IBPH produced a layered-depth image at this virtual camera position. We then converted the closest point on the hull into a 3D height map for analysis. We note that for IBPH, we turned off the spatial smoothing described in Section 4.7 in order to better evaluate the algorithm.

The results of these experiments are presented in Figure 17. We show the height field as well as the 3D error  $E_{3D}$  computed from each reconstruction. As one might expect, all reconstruction approaches benefited from using more (and more oblique) reference views. The visual hull for this configuration of cameras and silhouettes is pyramidal, as is apparent in the top row of the figure. Clearly, the image-based visual hull reconstruction deviates significantly from a plane, and thus the 3D error is high. Note that in the figure, the color map ranges from 0 to 6 for the IBVH reconstructions. The middle row of the figure presents the results from the GVC algorithm, which was executed using a  $160 \times 160 \times 45$  voxel space. The GVC (and IBPH) reconstructed surfaces exhibit cusps, which were described in Section 4.8. As reference views that are more oblique to the plane are included, the height of the cusps diminishes. The GVC reconstruction attained using all 24 reference views is quite close to planar. Note that the color map in the figure ranges from 0 to 2.5 for the GVC and IBPH reconstructions.

Finally, the bottom row of the figure presents the results of the IBPH algorithm. Each IBPH reconstruction is a big improvement over its corresponding IBVH reconstruction, as evinced in the lower 3D error. The 3D surface found using 8 reference views compared quite well to GVC. However, the surfaces generated using 16 and 32 reference views were noisier than those produced using GVC. This noise in the height fields results in a higher 3D error. The inaccuracy of the IBPH height fields compared to GVC is hardly surprising, since IBPH sacrifices some quality for speed. When computing photo-consistency, IBPH uses a small, fixed size footprint in each reference view that

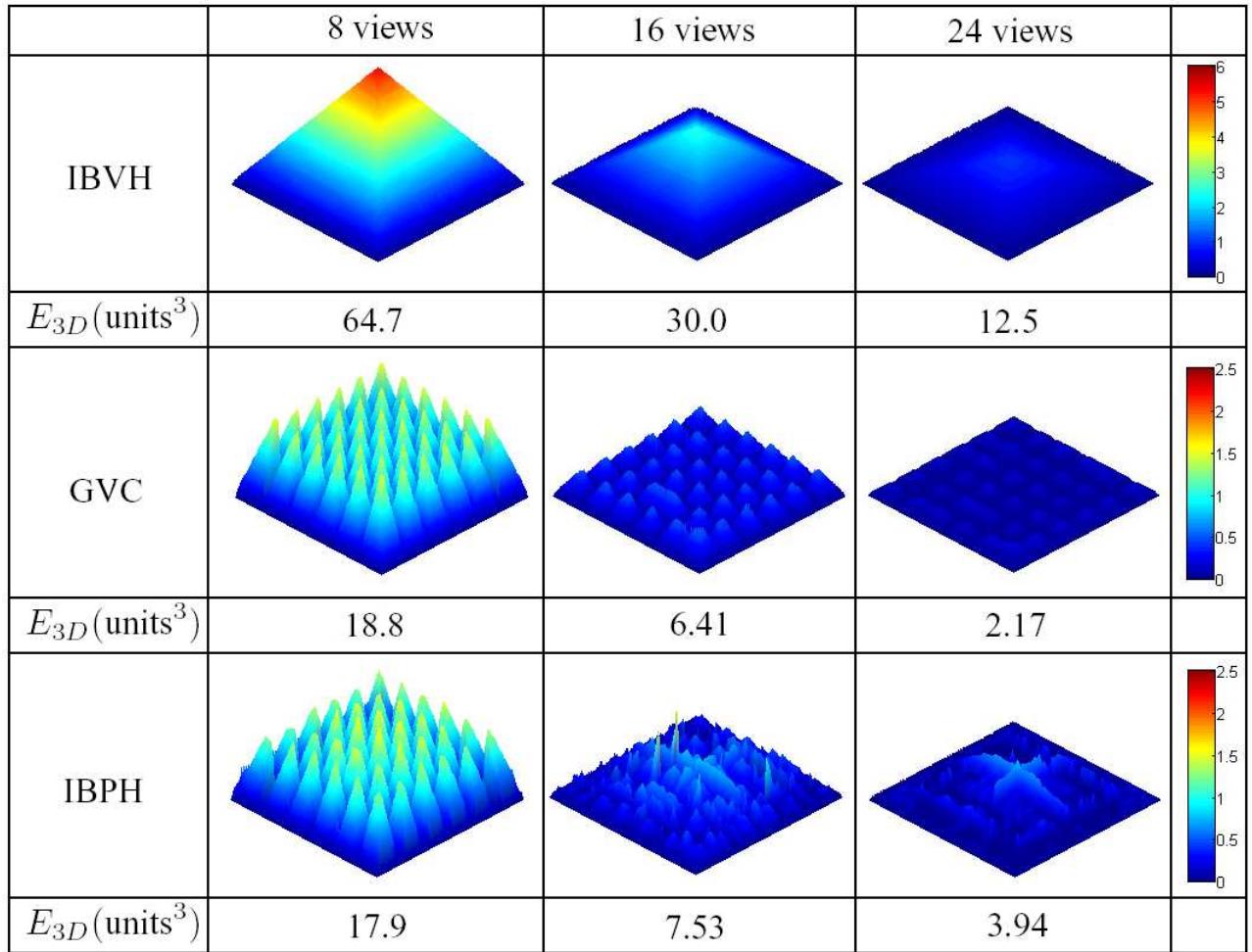


Figure 17: Reconstructions of the SynthPlane data set. From left to right: using 8, 16, and 24 reference views. From top to bottom: IBVH, IBPH, and GVC. See text for details.

is independent of the location of the reference view relative to the surface being reconstructed. It has been shown [20] that using such an imprecise approximation to the projection of a voxel can result in some geometric inaccuracy on the reconstructed surface. The IBVH/IBPH algorithms are conservative in their approach to computing visibility [7], while visibility in GVC (in particular, GVC-LDI [13]) is exact. Since visibility is part of the photo-consistency measure, imprecise modeling of visibility in IBPH also results in some geometric inaccuracy. Finally, recall that in IBPH we terminate the reconstruction before all rays are photo-consistent, as described in Section 4.6. In these experiments, we set  $M = 10$ . This also results in some geometric inaccuracies on the reconstructed surface. However, for the SynthPlane data set, the IBPH algorithm ran roughly two orders of magnitude faster than GVC.

## 6.2 2D Error

After performing a 3D reconstruction, we can render the reconstructed model to generate a synthetic image  $S$  at a new viewpoint  $V$ . We can then compare the pixels in  $S$  to the pixels in reference view  $R$ , taken at location  $V$  that is not a viewpoint for one of the reference views used to reconstruct the scene.

This 2D error measures the difference between  $S$  and  $R$ . Since  $S$  and  $R$  are color images, they have pixels with  $r$ ,  $g$ , and  $b$  components. Let the color components in the  $i$ th pixel of  $S$  be referenced as  $S(i).r$ ,  $S(i).g$ , and  $S(i).b$ , respectively, and likewise for  $R$ . We then define the new view synthesis error to be the mean square error between

$S$  and  $R$ ,

$$E_{2D} = \frac{\sum_{i=1}^M (S(i).r - R(i).r)^2 + (S(i).g - R(i).g)^2 + (S(i).b - R(i).b)^2}{M}, \quad (5)$$

where  $M$  is the number of pixels used in the comparison.

We generated an image  $R$  of the SynthPlane scene by rendering the multi-colored plane to a camera location  $V$  directly above the plane, for an elevation angle  $\phi = 0^\circ$ , and a radius 10.5 units up the  $z$  axis. This view is synthesized using the IBVH, GVC, and IBPH algorithms, for the three sets of images as before. The synthesized images, as well as their corresponding 2D error are presented in Figure 18. As seen in the top row of the figure, the new views synthesized using the IBVH algorithm have a large 2D error, which results from the pyramidal geometry of the visual hull.

The new views synthesized using the GVC and IBPH algorithms are shown in the middle and bottom rows of the figure, respectively. These synthesized views have significantly lower error due to the more accurate 3D geometry they reconstruct. Contrary to intuition, for the IBPH new views, the 2D error did not consistently decrease as more reference views were used. We demonstrated in Section 6.1 that using more reference views with the IBPH algorithm results in a more geometrically accurate reconstruction. However, the view-dependent texture mapping approach used by the algorithm becomes problematic when many reference views are used on a noisy surface, since the reference view being used to texture map a local point on the surface changes quickly along the surface. In the presence of inaccurate geometry, this results in small miscolorings, especially on the edges between two different colors as seen in Figure 18. GVC, in contrast, determines a color for a voxel by averaging all the pixels to which the voxel projects in each of the reference views. Such an averaging scheme could be used instead of VDTM in the IBPH approach, when helpful.

### 6.3 Noise Experiments

Finally, we present experimental results that analyze the system sensitivity to noise, both in the camera calibration and the image pixels.

First, we injected noise into the camera calibration of the SynthPlane data set, by displacing the 2D projection of each 3D point in a random direction, for each reference view. The length of the displacement was fixed for each experimental run. Multiple runs were performed for each length and the results averaged. The experimental results appear in the top row of Figure 19. To keep the presentation concise, we do not show the 3D height fields and 2D new view synthesis images, and instead present plots of the

3D and 2D error as a function of camera calibration noise. As expected, the both the 3D and 2D error increase with more camera calibration noise.

Next, we added white, zero-mean Gaussian noise to the pixels in each color plane of the reference views. After adding the noise, the value was clipped to the range [0, 255]. The standard deviation of the noise was fixed for each experimental run. Multiple runs were performed for each standard deviation and the results averaged. The experimental results appear in the bottom row of Figure 19, which presents plots of the 3D and 2D error as a function of pixel noise. As expected, the both the 3D and 2D error increase as the noise increases.

## 7 Conclusion

In this paper we have presented our Image-Based Photo Hulls algorithm that efficiently reconstructs views of the photo hull. Our algorithm extends IBVH by utilizing the color information in the reference views to further constrain the 3D space containing scene surfaces. The more accurate geometry reconstructed by our technique often results in better new views synthesized of the scene.

Our IBPH algorithm's efficiency comes from the following sources. First, it computes the photo hull starting from the visual hull. Using the IBVH algorithm, we are able to quickly find the visual hull, efficiently removing a large part of the 3D space that does not contain the surfaces being reconstructed. Second, our IBPH algorithm only computes that portion of the photo hull that is visible to the virtual viewpoint. Since this is the viewpoint that is being synthesized, it is not necessary to reconstruct that part of the 3D scene that is not visible in rendered view. Finally, we have demonstrated the effect of increasing the sampling size  $DX$  and  $DY$  to tradeoff accuracy for speed.

Our method inherits all the limitations of algorithms that reconstruct the photo hull. Scene surfaces must be sufficiently colorful in order to be properly reconstructed. The photo-consistency measure we use assumes a Lambertian scene. Reconstruction of significantly non-Lambertian scenes requires a more sophisticated approach for determining photo-consistency. We should note that visual hull reconstruction algorithms like IBVH do not have a problem with non-Lambertian scenes, as they do not perform color matching. The depth maps that are reconstructed with the IBPH algorithm, while geometrically more accurate, can be noisier than those of the IBVH algorithm. Finally, the IBPH algorithm is significantly slower than the IBVH algorithm.

## 8 Future work

There are several future directions we are interested in pursuing. The photo-consistency measure presented here does not take into account surface orientation or the dis-

	8 views	16 views	24 views
IBVH			
$E_{2D}$	6770	3776	1778
GVC			
$E_{2D}$	565.3	405.5	360.2
IBPH			
$E_{2D}$	522.3	742.8	691.1

Figure 18: Synthesized new views of the SynthPlane data set. From left to right: using 8, 16, and 24 reference views. From top to bottom: IBVH, IBPH, and GVC. See text for details.

tance of the reference view from the surface. Correlation-based matching similar to [19] might improve reconstruction quality. Additionally, we do not take into consideration temporal coherence. Motion constraints could be imposed to improve efficiency and reconstruction quality. Finally, an adaptive approach to determining  $|\Delta P|$ , the step size taken along each ray might improve the efficiency of our algorithm. When the photo-consistency measure is *very* inconsistent, one might take larger steps. As the consistency improves, one could take smaller steps until the ray becomes photo-consistent.

## 9 Acknowledgements

We thank Wojciech Matusik, Chris Buehler, and Leonard McMillan of MIT for sharing the IBVH source

code and system configuration with us. We also thank Bruce Culbertson, Tom Malzbender, and Irwin Sobel of HP Labs for several useful discussions regarding the IBPH algorithm. Finally, we thank Dan Gelb for producing the virtual conference room model used in Figure 14. This work has been funded by HP labs.

## References

- [1] Debevec P, Taylor C, Malik J. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach. Proceedings of SIGGRAPH, 1997; p. 11–20.
- [2] Prock A, Dyer C. Towards Real-Time Voxel Coloring. Proceedings of the Image Understand Workshop, 1998; p. 315–321.

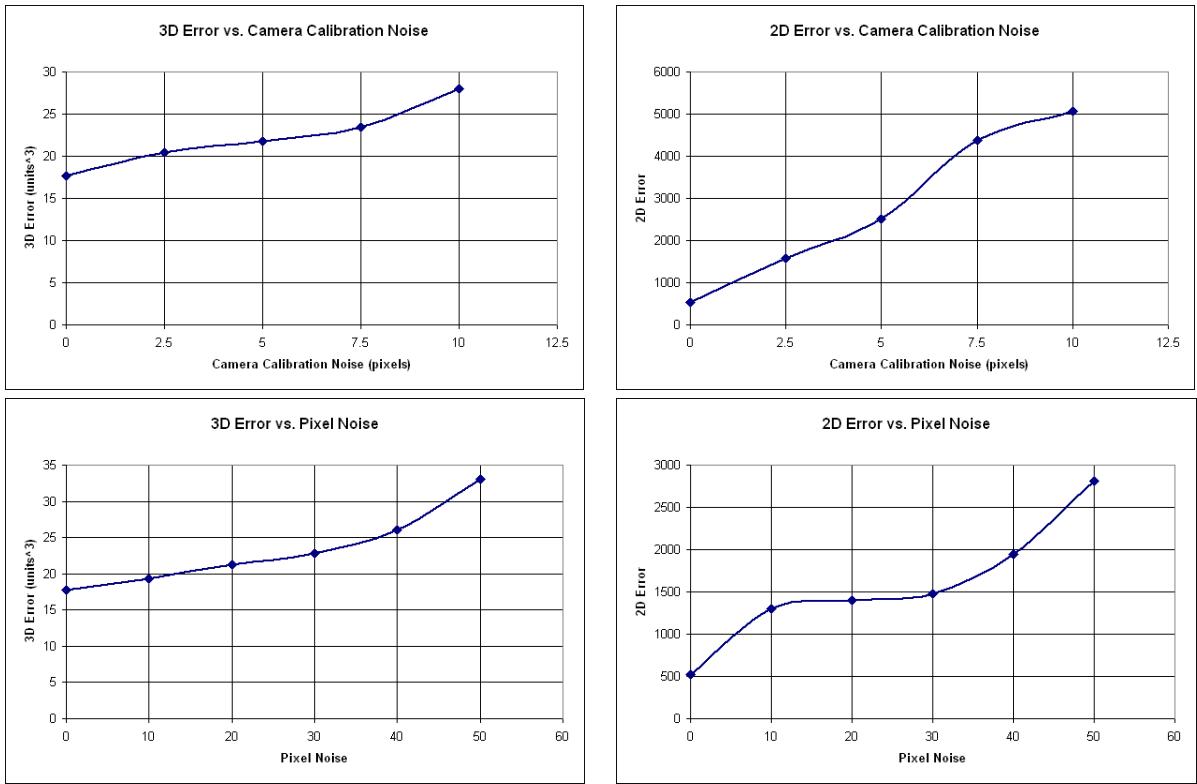


Figure 19: Noise analysis. The upper plots present the 3D and 2D errors as a function of increasing noise in the camera calibration, while the lower plots present the 3D and 2D error as a function of increasing noise added to the image pixels.

- [3] Narayanan P, Rander P, Kanade T. Constructing Virtual Worlds Using Dense Stereo. Proceedings of the International Conference on Computer Vision, 1998; p. 3–10.
- [4] Moezzi S, Tai LC, Gerard P. Virtual View Generation for 3D Digital Video. IEEE Multimedia 1997; 4(1):18–26.
- [5] Vedula S, Baker S, Seitz S, Kanade T. Shape and Motion Carving in 6D. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2000; p. 592–598.
- [6] Carceroni R, Kutulakos K. Multi-View Scene Capture by Surfel Sampling: From Video Streams to Non-Rigid Motion, Shape, and Reflectance. Proceedings of the International Conference on Computer Vision, 2001; p. 60–67.
- [7] Matusik W, Buehler C, Raskar R, Gortler SJ, McMillan L. Image-Based Visual Hulls. Proceedings of SIGGRAPH, 2000; p. 369–374.
- [8] Laurentini A. The Visual Hull Concept for Silhouette-based Image Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence 1997; 16(2):150–162.
- [9] Slabaugh G, Schafer R, Hans M. Image-Based Photo Hulls. Proceedings of the 1st International Symposium on 3D Data Processing, Visualization, and Transmission, 2002; p. 704–708.
- [10] Szeliski R. Rapid Octree Construction from Image Sequences. CVGIP: Image Understanding 1993; 58(1):23–32.
- [11] Kutulakos K, Seitz S. A Theory of Shape by Space Carving. International Journal of Computer Vision 2000; 38(3):199–218.
- [12] Seitz S, Dyer C. Photorealistic Scene Reconstruction by Voxel Coloring. International Journal of Computer Vision 1999; 35(2): 151–173.
- [13] Culbertson WB, Malzbender T, Slabaugh G. Generalized Voxel Coloring. Proceedings of the ICCV Work-

shop: Vision Algorithms Theory and Practice, 1999; p. 100–115.

- [14] Eisert P, Steinbach E, Girod B. 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; p. 3509–3512.
- [15] Broadhurst A, Drummond TW, Cipolla R. A Probabilistic Framework for Space Carving. Proceedings of the International Conference on Computer Vision, 2001; p. 388–393.
- [16] Dyer C. Volumetric Scene Reconstruction from Multiple Views. In: Davis L, editor. Foundations of Image Understanding. Kluwer; 2001, p. 469–489.
- [17] Slabaugh G, Culbertson WB, Malzbender T, Schafer R. A Survey of Volumetric Scene Reconstruction Methods from Photographs. Proceedings of Volume Graphics, 2001; p. 81–100.
- [18] Shade J, Gortler S, He L, Szeliski R. Layered Depth Images. Proceedings of SIGGRAPH, 1998; p. 231–242.
- [19] Faugeras O, Keriven R. Variational Principles, Surface Evolution, PDE's, Level Set Methods and the Stereo Problem. IEEE Transactions on Image Processing 1998; 7(3):336–344.
- [20] Steinbach E, Girod, B, Eisert, P, and Betz A. 3-D Reconstruction of Real-World Objects Using Extended Voxels. Proceedings of the International Conference on Image Processing, 2000; vol 1, p. 569–572.