Reconstructing specular objects with Image Based Rendering using Color Caching

by

Vikram Chhabra

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

____

May 2001

APPROVED:

____

Professor Mark R. Stevens, Thesis Advisor

____

Professor Michael A. Gennert, Thesis Reader

____

Professor Micha Hofri, Head of Department

Abstract

Various Image Based Rendering (IBR) techniques have been proposed to reconstruct scenes from its images. Voxel-based IBR algorithms reconstruct Lambertian scenes well, but fail for specular objects due to limitations of their consistency checks. We show that the conventional consistency techniques fail due to the large variation in reflected color of the surface for different viewing positions. We present a new consistency approach that can predict this variation in color and reconstruct specular objects present in the scene. We also present an evaluation of our technique by comparing it with three other consistency methods.

Acknowledgements

I would like to thank Mark Stevens, my thesis advisor, whose support, insight and direction made this work a success and for providing me with the all the resources and funding to work on this project; Micheal Gennert, my thesis reader, whose guidance and encouragement helped me to formulate my ideas and for proof-reading this work; the WPI Computer Science department for providing me with this opportunity to work; and above all, my parents who made all this possible for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computer Graphics has come a long way from rendering three dimensional polygons using perspective to high quality rendering using advanced techniques such as ray tracing and radiosity. Many complex tools have been developed to create models and render images using Computer Graphics, but modeling complex scenes is a hard task and rendering them is computationally demanding. As a result, producing photo-realistic images is a difficult task. Moreover, the triangle primitive seems to be insufficient to support the increasing complexity of the models.

Imagine how difficult it could be to model and render an image of a huge complex structure such as the Statue of Liberty using Computer Graphics. One would start from studying the blueprints of the structure to understand the geometry, then figure out the appropriate lighting model for illumination and still it may not look photo-realistic because the rendering will be too synthetic looking. Such rendered images cannot account for the natural coarse appearance or the dirt on the structure.

What if one could model and render images of a structure just by taking pictures of it with a camera. Image Based Rendering (IBR) provides such a methodology. IBR describes a set of techniques that allow three dimensional graphical interaction with objects and scenes whose original specification began as images or photographs.

1.1 Context and Motivation

Image Based Rendering is the fusion of Computer Graphics and Computer Vision. Computer Graphics generates images from a model using its mathematical description such as dimension, camera parameters, lighting, etc. Computer Vision does the opposite; it infers the shape and surface properties of the object from its images. Given a set of views of a scene, Image Based Rendering reconstructs the three dimensional data structure of the scene and uses it to render views of the scene from new viewpoints.

Image Based Rendering offers a vast range of techniques to model and render scenes using images. IBR has grown enormously over the past six years, trying to overcome the various issues with different techniques. One remaining problem is the reconstruction of specular objects.

Specular objects are those that are made of reflective surfaces, such as polished wood, and hence show a large variation in color under different viewing orientations. This implies that the same object can project different colors in images taken from different viewpoints. The color of the pixel in the image is the central source of information to many IBR algorithms. Specular highlights mislead this information and cause the IBR algorithms to fail.

In this work, we study the behavior of specular objects and designed a new approach that predicts such variations in the color of specular objects and hence prevents the IBR algorithm from failing.

1.2 Problem Overview

IBR techniques such as Generalized Voxel Coloring (GVC) [1] take a set of images of a scene along with their camera parameters as input to the system. There is no information about the lighting, surface properties or 3D configuration of the scene. Using the input, GVC generates a 3D model of the scene and renders images from new viewpoints. The consistency between the surface color projected in all the images of the scene is used to generate the 3D model. Specular highlights in the input images lead to false information about the surface and hence cause the algorithm to fail. Figure 1.1 shows a torus that has a specular highlight which causes GVC to fail. The GVC reconstruction is shown in Figure 1.2. This failure is due to the large variation of surface color in different images and making the surface inconsistent among the different input views. For this thesis we have developed a new approach that can predict this variation in surface color and reconstruct specular objects.

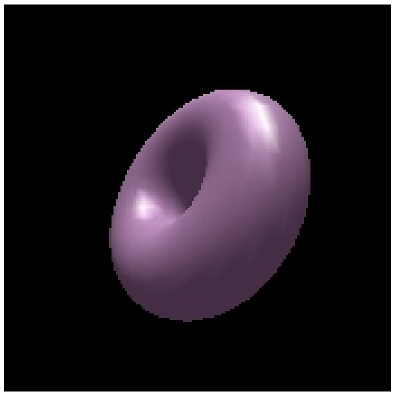
Figure 1.1: A torus with bright specular highlight

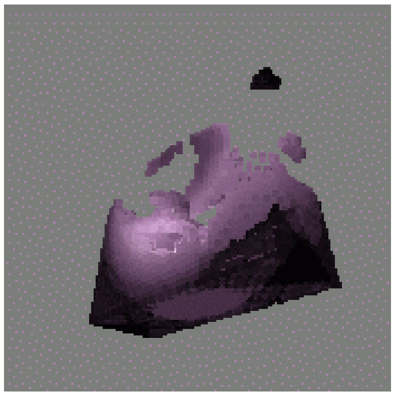
Figure 1.2: GVC fails to reconstruct the specular highlighted area of the torus

1.3 Thesis Structure

An overview of the context of this work is presented in Chapter One. Chapter Two talks about various related techniques used to reconstruct scenes from images, with their advantages and limitations. Chapter Three defines the problem by presenting our study of specular objects and reasons why IBR algorithms fail to reconstruct specular objects. Chapter Four presents our approach to deal with specularity. Chapter Five contains the results and evaluation. Chapter Six summarize the conclusions.

Chapter 2

Previous Work

2.1 Techniques proposed

Various techniques have been proposed for volumetric scene reconstruction. These vary from full 3D representations to 2D interpolations. Attempts have also been made to work with weakly calibrated cameras, since calibrating cameras, or finding the extrinsic and the intrinsic parameters, can be a difficult task to handle. The following subsections discuss some of these techniques. For a complete literature survey see [7].

2.1.1 Voxel Coloring

Seitz and Dyer have presented the problem of scene reconstruction as, given a set of input images and a 3D space V, determine the subset $S \subset V$ which is *color consistent* with the input images [6]. The Voxel Coloring algorithm works by discretizing scene space into a set of voxels that is traversed and colored in a special order. Only those voxels that are consistent are retained. A voxel is consistent if it projects to the same color (within a permissible error, which is defined by threshold,) on all the images from where it is visible.

To avoid multiple passes through the voxel space, this technique introduces a constraint known as the ordinal visibility constraint [6]. According to this constraint, the voxel space is partitioned into a series of layers of increasing distance from the camera. Then each layer is traversed one by one, passing a plane through a volume. This helps to determine the occlusions between voxels and hence their visibility information. Only those voxels that are visible are colored. The algorithm's complexity is relaxed so that only a single pass of the volume is required. This places a restriction on the camera locations, which is a significant limitation. The algorithm pseudo code is shown in Figure 2.1.

```
for each layer in the voxel space {
   for each voxel in that layer {
      project the voxel to each image
      collect the set of pixels to which this voxel projects
      evaluate this voxel's consistency
      if (consistency < threshold) {
         color this voxel
      }
   }
}
```

Figure 2.1: Pseudo code for Voxel Coloring

Voxel consistency is defined as the standard deviation in the color of the set of pixels in all images, on which a voxel projects. So a voxel will be colored only if the consistency is less than the threshold value.

The algorithm has a linear time and space complexity with respect to number of images. It traverses each voxel exactly once, hence it is of the order 0(N), where N is the number of voxels. It was tested on both real and synthetic images. Holes in the reconstruction were only visible from non-basis viewpoints. Another important fact that must be stated about this technique is that the cameras need a very high precision of calibration, which is problematic.

2.1.2 Space Carving

The space carving theory [3] reconstructs arbitrary shaped scenes from a set of photographs with no constraints on the camera placement. It finds a set of all *photo-consistent shapes* from the given set of input photographs and tries to compute a shape from this set, known as the photo hull, a maximal shape. The only requirements are that the viewpoint of each photograph is known in a common 3D world reference frame and the scene radiance follows a known locally computable radiance function (locally computable radiance function is defined as a special class of scenes for which global illumination effects such as shadows, transparency and inter-reflections can be ignored).

A concrete characterization is provided for the family of scenes that are photo-consistent with the set of input photographs. This characterization is defined by a background and a radiance constraint. The background constraint states that if V is the 3D scene being constructed and viewed under perspective projection, then no point on V projects to a background pixel. If a photograph taken from a position c contains identifiable background pixels, this constraint restricts V to a cone defined by c and the non-background pixels in the photograph. Given N such photographs, the scene is restricted to a visual hull, which is a volume intersection of their corresponding cones. Though this constraint is good enough to reconstruct the visual hull, it fails when there are no background pixels or the background pixels are difficult to identify. The radiance constraint takes advantage of the fact that the scene belongs to the class of locally computable radiance models. Hence given an *a priori* computable radiance model for the scene, it can be determined whether or not the given shape V is photo-consistent with the given set of input photographs.

The reconstruction is done by carving an arbitrary volume such that it converges to the photo hull. The algorithm makes two assumptions, first that the initial volume contains the true scene and second that the surface of the true scene conforms to a radiance model defined by the consistency check algorithm. The space carving algorithm pseudo code is shown in Figure 2.2.

```
Initialize V to the volume containing the true scene.
for all voxels on the surface of the volume {
   project each surface voxel on to the photographs
   determine the photo-consistency(pixels color,
         optical rays to the corresponding optical center)
   if (voxel is non-photo-consistent) {
      carve it
   }
   loop until no more voxels are carved.
}
```

Figure 2.2: Pseudo code for Space Carving

The total number of consistency checks is bounded by N x M, where N is the number of input photographs and M is the number of voxels in the initial volume.

Multiple passes are swept through the volume to make sure that voxels are photo-consistent with all the images. During each pass a plane is swept in increasing or decreasing direction of x,y or z, hence making a total of six passes. A threshold is used on the standard deviation of the pixel to decide whether or not to carve the voxel. This is the same consistency test used in Voxel Coloring [6]. The advantage of this technique over the Voxel Coloring [6] is that it does not place any constraint on the camera position.

2.1.3 Generalized Voxel Coloring

Generalized Voxel Coloring (GVC) [1] is an extension of the Voxel Coloring [6] and Space Carving [3] algorithms. Unlike voxel coloring, GVC removes any constraint on the position of the cameras. On the other hand, it uses all the images to evaluate the consistency of a voxel unlike space carving which uses only a subset of them. The key difference is the way visibility of a voxel is determined. Voxel coloring places a constraint on the camera position and sweeps the volume in one direction so that a visibility of the voxels that might occlude a voxel are already known. Space carving uses multiple scans to remove the constraint on the camera position. Though Space Carving never carves a voxel it should not but its conservative approach may leave some voxels that should have been carved.

GVC has two variants, one is computationally expensive and other consumes lot of memory. The basic version of GVC maintains a list of all surface voxels and after every carving iteration, this list is computed again as the visibility information changes. This makes it computationally expensive. The second version called GVC-LDI avoids the computation by storing the volume information in advance, but this requires a large amount of memory.

The basic GVC algorithm assigns a unique id to each voxel. A list of all visible voxels is created and defined as Surface Voxel List (SVL). Each voxel on this list is tested for consistency and every time a voxel is carved the SVL is re-evaluated. The algorithm pseudo code is shown in Figure 2.3.

Observe that the algorithm is very similar to Voxel Coloring [6] and Space Carving [3], only the visibility computation is different.

The second version maintains a two dimensional SVL, which means that each voxel in the SVL has a sub-list that keeps track of the next visible voxel if this one was carved. The advantage is clear that the SVL now need not be reevaluated. But

```
initialize SVL
for every voxel V on SVL {
   compute V's consistency
   if (V is inconsistent) {
     carve V
     reevaluate SVL
   }
   loop until no more voxels are carved
}
```

Figure 2.3: Pseudo code for Generalized Voxel Coloring

this requires a large amount of memory.

The performance of this technique was compared to the space carving approach using two scenes. The basic GVC takes about double the memory as compared to the Space Carving. The LDI takes about five times more memory than the original GVC. The number of consistency checks are lower for GVCs as compared to the Space Carving.

2.1.4 Volumetric Warping

Slabaugh *et al.* have presented a technique to reconstruct large-scale scenes using Volumetric Warping [7]. This approach is an extension of the voxel based techniques such as Generalized Voxel Coloring [1] with spatially adaptive voxel size that increases away from the cameras. Warping the voxel space allows an infinitely large scene space to be modeled with finite number of voxels. A warping function is defined such that no voxels overlap and no gaps are left between the voxels.

The voxel space is divided into two regions: the interior region to model foreground objects and the exterior region to model background surfaces. The volumetric warp does not affect the volume in the interior space, providing backward compatibility with other voxel based techniques. A frustum warping function is defined which warps the exterior voxels in proportion to the distance of the voxel from the interior space. The frustum warp assumes the both the interior and exterior space have rectangular shaped outer boundaries. The outer regions that are to be warped are labeled as $\pm x$, $\pm y$ and $\pm z$. For each region a warping function is defined as:

$$x_w = x\frac{x_e - x_i}{x_e - |x|} \tag{2.1}$$

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, x_i is the distance along the x-axis from the center of the interior space to the outer boundary of the interior space, x is the pre-warped x-coordinate and x_w is the x-coordinate after warping, as shown in Figure 2.4. Thus, a point on the boundary of the inner volume does not move, whereas the point on the boundary of exterior space is warped to infinity. The equivalent y-coordinate for this x_w is found using the equation of the line:

$$y_w = y + m(x_w - x) \tag{2.2}$$

where m is the slope of the line connecting the point (x, y) with the point a, as shown in Figure 2.4.

Reconstructing the scene using warped reconstruction volume is a difficult task. The cameras are embedded in the volume due to which no voxel is visible in more than one camera. To avoid this situation, the authors pre-carve a section of the volume initially so that each surface voxel is visible by at least two cameras.

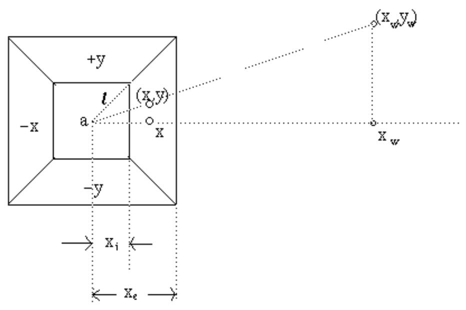

Figure 2.4: Finding the warped point

2.1.5 Shape reconstruction in Projective Grid Space

Saito *et al.* have focused more on the problem of camera calibration rather than scene reconstruction [5]. Camera calibration is a very difficult task. This technique reconstructs scenes using weakly calibrated cameras. This is achieved by defining a projective grid space, which uses two basis views and a fundamental matrix relating these views. Once the projective grid space is formed, a projective shape can be reconstructed from all images of the weakly calibrated camera [5].

The projective grid space is constructed by selecting two images as the basis of the projective grid space. Each pixel in the first image defines a grid line in space. The rays coming out of image two intersect the grid line and define the grid node points in the space. These node points are recorded by the horizontal displacement or vertical displacement on the second image. This is possible since the fundamental matrix limits the position on the epipolar line in the second image. The epipolar line, on the second image, is the projection of the grid line in space formed by the first image. This way the projective grid space is defined where each node is represented by (p,q,r) where (p,q) is the pixel position of the grid line in the first image and r is the horizontal (or vertical) displacement on the second image of the intersecting grid line, passing through the epipolar line.

Now, the relationship between the projective grid space and any arbitrary image is determined by the two fundamental matrices of the image with the two basis images. Hence, every grid point can be projected onto every image by using only the fundamental matrices between the image and the two base images.

The concept of projective grid space is used to avoid the heavy calibration required in the Voxel Coloring [6]. Voxel Coloring requires that the geometric relationship between every voxel and image pixel be known. By applying the projective grid concept, the Euclidean calibration can be replaced by the projective calibration, which is sufficient for scene reconstruction since every grid point's relation to each image is now known.

Experiments were performed using this approach on a basketball scene taken by a 3D Video Dome System. Out of the 51 images, two, whose optical axes were nearly perpendicular, were chosen as the base images. The calibration was done by placing LED point lights in a straight line, since the LEDs will be visible from all images. About 500 correspondence points are obtained using the LEDs and the fundamental matrices are estimated using Zhang's method [13]. About 320x240 epipolar lines are projected from base image one onto base image two for generating the projective grid space. And 320 grid points are defined for each epipolar line by the horizontal position in base image two. This makes a total of 320x240x320 grid points in the space. In any reconstructing technique, occlusions cannot be interpolated using an image-based method until the 3D structure is known. The reconstructed projective shape provides a dense correspondence map between any two images and this can be used to synthesize new intermediate image viewpoints taking occlusions into account.

2.1.6 Silhouettes

Silhouettes are one of the most basic cues to 3D structure, and despite their simplicity they have been useful in a wide variety of image-based modeling and rendering research. Szeliski has used object silhouettes from a video stream of a rotating object to reconstruct its geometry [9].

The 3D volumetric description is recovered from the binary silhouettes of the object against its background from multiple views. Each image is converted into a binary silhouette by differencing the image with an empty background. Then each cube in an octree volumetric 3D model is projected (using the known camera position) into the silhouette and those cubes that fall outside the silhouette are removed. If a cube falls partially on the silhouette, it is marked for later subdivision, and the process is repeated until a minimum resolution is achieved.

By using more views of the object, this technique can recover the visual hull of the object. For many objects the visual hull is same as the shape itself, although for shapes with a complicated concave structure, some of the volume will remain.

2.1.7 Real-Time Voxel Coloring

Prock and Dyer have proposed three methods for speeding up the original Voxel Coloring algorithm [6] so that it runs in real-time. The first one uses hardware texture mapping, the second implements a multi-resolution approach and the third utilizes the fact that dynamic scenes are temporally coherent [4].

Voxel Coloring [6] places constraints on the camera position and scans the volume layer by layer. Then in every layer, the voxels are projected onto the images. Prock and Dyer suggest that instead of projecting the voxels onto the images, project the images on the plane of voxels [4]. This can be achieved using *"hardware texture*

```
prewarp all images
for each voxel layer x {
   for each image y {
      texture map layer x with image y
      for each voxel in layer x {
         store its color value
      }
   }
   for each voxel V in layer x {
      if (V's colors are correlated) {
         color voxel V
         update image pixels to reflect occlusions
      }
   }
}
```

Figure 2.5: Pseudo code for Real-Time Voxel Coloring

mapping". The images must be pre-warped for the transformation from the world to image coordinates to correctly follow the pinhole camera model. The pseudo code for this algorithm is shown in Figure 2.5.

An interesting change in this algorithm, as compared to the original Voxel Coloring algorithm [6], is that the computation is now measured on a per-image basis and not on a per-voxel basis. The occlusion information is stored in the images and this information is updated for every layer traversed.

The second approach proposed is a multi-resolution approach - moving from *"coarse to fine coloring"* [4]. In this technique, the coloring of the scene is started at a low resolution, so as to speed up the process. Voxels are then added to the original low resolution voxel set. These new voxels are divided into smaller units and colored according to the standard algorithm. The problem with this approach is that while coloring at low resolution, those voxels which are partially colored are rejected. This can lead to the formation of holes in the reconstruction. To fix this problem, the authors suggest a nearest neighbor search strategy to find the missing voxels. This strategy utilizes the fact that the surfaces are spatially coherent.

The final method suggested is *"dynamic voxel coloring"* [4]. Dynamic scenes, like a video clip, are temporally coherent. So a scene-space that was found to be empty in a previous frame, need not be processed in the current frame. Though this optimization will speed up the coloring process, but it fails in the case when the scene changes suddenly, such as the appearance of new objects. A search strategy is implemented to restore the missing voxels. It is similar to the one used for the coarse to fine approach, but needs a threshold to be set, which would define how fast the scene changes. This will help to determine how fast the surface completely moves out of the view window. Every time a frame is colored, a seed color is updated to reflect any changes due to the motion. Once the seed color is augmented, the voxels are subdivided into smaller units. If changes are made to the new subdivided voxels, their seed colors are accordingly updated.

The experimental results for this approach show improvement in the performance of the original algorithm. The pre warp images help to speed up the process, as compared to Tsai's camera calibration [10], which is computationally expensive. The pre-warping of the images can be done before the actual coloring of the scene. The texture mapping method gave modest performance for scene resolutions up to 160 voxels. The reconstruction obtained from texture map optimization had the object colors mixed with the background. This is because of the interpolation of colors between the foreground and background pixels. The multi-resolution approach shows a good improvement in performance with a speedup of 1.2 times with low resolution and 41.1 times for higher resolution. The dynamic approach was tried on a 3 frame scene. The original coloring took 3.70 seconds, whereas with dynamic coloring, this figure dropped down to 2.74 seconds.

2.1.8 Building Models from Range Images

All the techniques discussed until now have used regular camera images. Levoy *et al.* [2] proposed a method for reconstruction from range images. Range images are images that have depth information for the scene. These images are taken using range scanners. The depths values are stored on a regular sampling lattice. A range surface is created by connecting the near neighbor points in a triangular fashion.

Levoy *et al.* [2] have defined a continuous implicit function, $D(\mathrm{x})$, represented by samples. This function defines the weighted signed distance of each point x to the nearest range surface along the line of sight of the sensor. A number of range images are taken and the signed distance function and the corresponding weights are computed for each. These are then used to compute the cumulative function $D(\mathrm{x})$ and cumulative weight $W(\mathrm{x})$. Finally, a zero crossing iso-surface is extracted from these cumulative functions corresponding to $D(\mathrm{x}) = 0$. The weights are computed using the dot product between each vertex normal and the viewing direction [8, 11]. To avoid the surfaces of the opposite side of the object, the weight function is tapered off behind the surface. The algorithm begins by setting all voxels to zero weight. Then range surfaces are formed for each image by constructing triangles. The weight at each vertex is computed and the signed distance function is determined by casting a ray from the sensor through each voxel near the range surface and then intersecting it with the triangle mesh. The weight inbetween the triangles is computed by interpolating the weights at the vertices. By substituting the weights in the computation equations [2], the zero crossing iso-surface is extracted from the volumetric grid.

Like some other reconstruction algorithms, this one too leads to formation of holes for the unseen regions. An extension to the above discussed algorithm is given by the authors for filling these holes. The points in the volume are classified as either unseen, empty or near the surface. Holes are the frontiers between unseen and empty regions. So surfaces placed at these frontiers can plug the holes. The modified algorithm initializes all voxels as unseen and update the voxels as discussed above. Then from the observed surface, following the line of sight backwards, each voxel is marked as empty. The iso-surface is extracted at zero-crossings and an additional surface is inferred between the empty and unseen regions.

This volumetric integration [2] approach was used to reconstruct a drill bit. The Zippering method [11] fails to reconstruct the object, but the volumetric approach generated a good watertight model. Like the voxel-based techniques, this method is not sensitive to color variations since it uses the 3D data from the range images for reconstruction instead of depending on color consistencies. A limitation of this algorithm is its inability to reconstruct sharp corners. Though the hole filling extension does fix this, the original algorithm needs to be improved to work without any hole filling mechanism. Thin surfaces also create a problem for this algorithm, since the distance function of the scans generated from opposite sides of a thin surface interfere with the front side.

2.2 Camera Calibration

Most volumetric reconstruction algorithms require camera calibration parameters. The work in this thesis too requires these parameters to project the 3D voxel onto the 2D camera image. In this section we explain how these camera calibration parameters are computed.

Camera calibration is the process of determining the intrinsic and extrinsic parameters of the camera. The intrinsic parameters include the internal camera geometric and optical parameters such as the focal length, lens distortion coefficient, uncertainty scale factor for x (due to TV camera scanning and acquisition timing error) and the computer image coordinate for the origin in the image plane. The extrinsic parameters refers to the positional orientation of the camera with respect to the world coordinate system. These can include the three Euler angles and three translation parameters.

These parameters are computed based on a number of points whose object coordinates in the world coordinate system are known and whose image coordinates are measured [10]. Once these parameters are determined, any point's coordinate can be transformed from the world coordinate system to the computer image coordinate system. This transformation is a four-stage process:

1. The point is transformed from the world coordinate system to the 3D camera coordinate, by a simple rotation and translation.

2. Then from the 3D-camera coordinate it is projected onto the ideal image coordinates using perspective projection with the pinhole camera geometry.

3. This image coordinate point is adjusted by the radial lens distortion parameters.

4. Finally, the real image coordinates are converted to computer image coordinates.

The intrinsic parameters are used for the computations of step 2 to 4 and the extrinsic for step 1.

2.3 Photo-Consistency Measure

All optical volumetric reconstruction techniques rely on a *consistency test* to determine when to remove a voxel from the volume. In this section, we discuss a common consistency test used by many reconstruction techniques [6, 3, 1].

Photo-Consistency means the same color is seen for the same surface in different images taken from different viewpoints. Voxel-based IBR algorithms work by matching the pixel color of the same object in the different images. Hence, the voxel consistency measuring method is an important factor that affects the quality of the reconstructed model.

Sietz and Dyer [6] state the problem of consistency as to determine the set of all pixels, from the given n images, π_i, onto which the same voxel, V, projects:

$$\cup_{1\ to\ n}\{\pi_i\} \tag{2.3}$$

where π_i is the set of pixels in image i which fall under the voxel V's projection, and compute the standard deviation, σ_V, of their pixel colors to evaluate the voxel V's consistency. If this σ_V is less than a specified threshold value, λ, then the voxel is declared as consistent:

$$\sigma_V \leq \lambda \Rightarrow voxel\ is\ consistent \tag{2.4}$$

$$\sigma_V > \lambda \Rightarrow voxel\ is\ inconsistent \tag{2.5}$$

Consistency evaluation must be monotonic test. A voxel that is once declared as inconsistent cannot become consistent again. Kutulakos and Seitz [3] proposed a property for color consistency as given two sets of pixels S and S' with $S \subseteq S'$, if S is inconsistent, then S' is inconsistent.

If a voxel is incorrectly carved, not only is a single voxel carved but all voxels that are occluded along a ray from the image causing the inconsistency are carved too. Hence, the consistency measure should be accurate to never carve a consistent voxel. The following subsection discusses one such method that requires least tunable parameters and can generate neat reconstructions.

2.3.1 Histogram Consistency Measure

Slabaugh *et al.* have discussed the issues of the consistency test used in the voxel-based methods such as GVC [1] and have presented a new approach based on *Histograms* that does not require any tunable parameters [7]. Instead of pooling the pixels from all the views of a given voxel, this method uses a series of paired tests. For any carving iteration, the set of pixels that project to a given voxel for a given image is defined as:

$$V_i^t \tag{2.6}$$

where V is the voxel, t is the iteration number and i is the image number from which the pixels have been projected. Hence, the consistency of the voxel is defined as a paired test:

$$consist(V_i^t, V_j^t) \; where \; i \neq j \tag{2.7}$$

This paired test returns true if the voxel is consistent for the pair of images i, j.

The histogram approach divides the RGB color space into 512 (8 x 8 x 8) uniform cubes or sub-spaces with each space 32 x 32 x 32 pixels wide. Each color value is encoded by number ranging between 0 and 7. For instance, the red color values are encoded as 0 for 0 to 31, 1 for 32 to 63 and so forth. Same is done for green and blue color. So a sub-space, defined as bin, in the color space can be identified by a triple number ranging between 0 and 7. For each voxel, a histogram is generated that keeps the bin count for all the color values projected on the image. Once the color values are encoded in the histogram, the voxel is tested for its consistency as follows:

$$consist(V) = \forall_i \forall_j Hist(V_i) \bigcap Hist(V_j) \neq \emptyset \; where \; i \neq j \tag{2.8}$$

where $Hist(V_i)$ are the set of pixels projecting on the voxel from i^{th} image. Similarly, $Hist(V_j)$ are the set of pixels for the j^{th} image. A voxel is declared as consistent if there is at least one none null intersection of bins among all mutually exhaustive combinations of the images. To avoid inaccurate decisions due to pixel values falling on boundaries on the bins, the boundaries are blurred, which means that a pixel value which falls near a boundary is considered to be present in both color spaces. Since the color space is in 3D, there can be case where a pixel falls under multiple sub-spaces or just a single sub-space.

The leniency in the consistency check, of having only one match in the color space to declare any two views consistent, does not limit the performance of the algorithm. This consistency measure was used with the GVC [1] technique and various scenes were carved using this technique [7].

2.4 Discussion

A number of techniques have been discussed in this chapter. These techniques vary in the following ways:

1. The method by which the visibility of the voxel is determined, i.e., whether the voxel is visible, carved or occluded.

2. The technique of how the images are calibrated. Some require accurate camera calibration while others can work with weakly calibrated cameras.

3. The measure of consistency, i.e., whether a voxel should be carved or not.

Though work has been done on former two, i.e., the visibility problem and camera calibration, not much work has been done on the issue of consistency measure.

Conventional consistency checks fail to reconstruct specular or textured objects. Voxel Coloring [6] uses the standard deviation in the pixel color value to determine the consistency of the voxel. Hence, in the lamp shade scene with textured surfaces as shown in Figure 2.6, the deviation would be low for the cream base with mostly uniform color, where as the deviation would be high for the textured red, green and blue leaves. But for both the regions a common threshold is used, so if the threshold value is too low then an incomplete reconstruction will be obtained, and if the threshold value is too high, then a noisy reconstruction will be obtained. Similar is the case for specular objects. Specular objects show high variation in color when viewed from different viewpoints as shown in Figure 2.7, and such high variation in color leads to a large deviation that exceeds the threshold value and gets over-carved. Hence more work is required to make these voxel-based techniques robust so that they can handle all kinds of scenes.

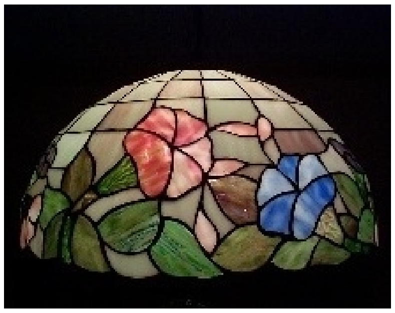
Figure 2.6: The textured leaves have high deviation as compared to cream base of the lamp shade

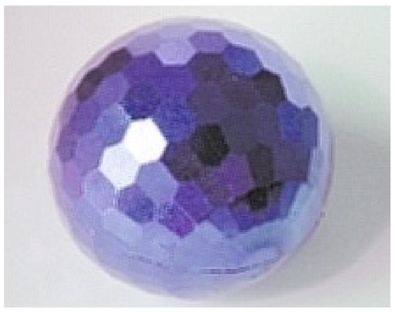
Figure 2.7: High variation in color due to different viewing angles accounts to a large deviation

In the following chapters, we shall discuss how specular objects behave and what are the problems associated with them. Then we present our consistency method to reconstruct specular objects.

Chapter 3

Problem Analysis

To understand the behavior of specular objects, it is important to study the lighting model used to define the color of surfaces. In 1975, Phong Bui-Tuong introduced the Phong shading and since then it has become the de facto standard for the three dimensional computer graphics [12]. There are two separate considerations for coloring a pixel. The first is the theoretical framework that calculates the light reflected at any point on the surface and second uses this framework to find the light intensity at a pixel onto which the polygon projects. The former is known as the *Local Reflection Model* and the latter as the *Shading Algorithm*.

The local reflection model introduced by Phong evaluates the intensity of the reflected light as a function of the orientation of the surface at the point of interest with respect to the position of a point light source and surface properties. It considers only direct illumination and behaves as if the object is floating in free space. Hence, any interactions with other objects, which could have resulted in shadows or inter-reflections, are not taken into account. This model simulates three types of surface reflections:

- Perfect Diffuse Reflection: these are matte surfaces, which reflect light equally in all directions (as shown in Figure 3.1) [12].

- Perfect Specular Reflection: these surfaces follow the law of simple reflection (i.e., the angle of incidence is same as the angle of reflection). So a thin sharp light will be reflected perfectly back without any divergence (as shown in Figure 3.2) [12].

- Imperfect Specular Reflection: most of the surfaces which exhibit highlights are not perfect mirrors, hence they are treated as a huge number of microscopic perfect mirrors with slightly different orientations. As a result, the reflection is spread over an area of the surface and forms a lobe (as shown in Figure 3.3) [12].

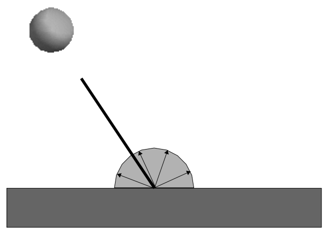
Figure 3.1: Perfect Diffuse Reflection

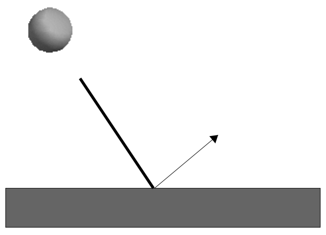
Figure 3.2: Perfect Specular Reflection

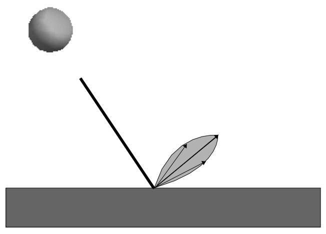
Figure 3.3: Imperfect Specular Reflection

The Phong Reflection model describes the color of a surface as a combination of three components. These are the ambient, diffuse and specular component. The ambient component is used to account for all the immeasurable indirect illumination in a scene. Consider a surface that is in between an object and the light. The object is not directly illuminated but is still visible. In such case, the ambient term makes sure that the object is still visible and not rendered as black. The diffuse component creates the uniform reflection in the direction from the surface but depending on the direction of light. Hence, this component is not a constant term like the ambience term. Finally, the specular component is responsible for the highlights seen on the surface. It is a view dependent term, which means that the specular highlight is visible only for certain viewpoints. This term can be tuned to reflect the light sharply or dispersed and hence can render almost all real life sceneries. The equation of light is:

$$ReflectedLight = I_a K_a + I_d K_d + I_s K_s \tag{3.1}$$

where I_a, I_d and I_s define the ambient, diffuse and specular component of the light source respectively and K_a, K_d and K_s define the ambient, diffuse and specular material properties respectively, such that:

$$K_a + K_d + K_s = 1 \tag{3.2}$$

The diffuse component of light, I_d, can be expanded to:

$$I_d = I_i \cos(\theta) \tag{3.3}$$

where θ is the angle between the surface normal $\mathbf{N}$ at the point of interest and the direction of light $\mathbf{L}$ and I_i is the intensity of the incident light. Similarly, the specular component can be expanded to:

$$I_s = I_i \cos^n(\omega) \tag{3.4}$$

where ω is the angle between the viewing direction $\mathbf{V}$ and the mirror direction $\mathbf{R}$ (as shown in Figure 3.4) [12]. Hence, the equation of light is expressed as:

$$I = I_a K_a + I_i(K_d \cos(\theta) + K_s \cos^n(\omega)) \tag{3.5}$$

$$I = I_a K_a + I_i(K_d(\mathbf{L} \cdot \mathbf{N}) + K_s(\mathbf{R} \cdot \mathbf{V})^n) \tag{3.6}$$

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{3.7}$$

where

L : Vector pointing to the direction of light

N : Surface normal at the point of consideration

R : Mirror direction of the surface

V : The viewing direction

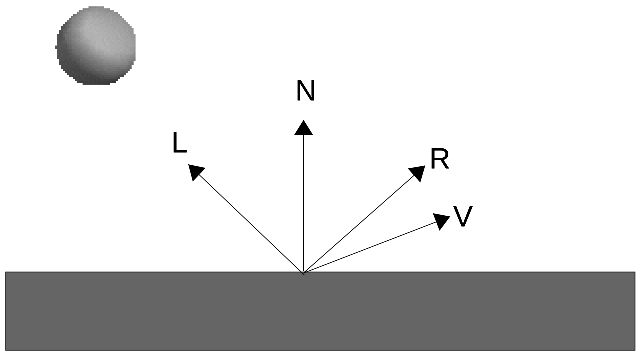

Figure 3.4: The Phong Model

3.1 Analyzing the Equation of Light

The voxel-based IBR techniques take images and camera parameters as input to the system. There is no information about the light source or the surface properties for the scene. Voxel-based IBR methods rely on color variation of the same surface in different images to reconstruct scenes. The color reflected by a surface depends on the viewing direction, surface normal, surface reflectance properties and the direction of light (Equation 3.6). Hence there can be a large variation in color of the surface when viewed from different viewing directions, even when the other parameters are kept constant. Such variation in color can cause the voxel-based IBR methods to fail. This variation can be computed if the surface properties and the light parameters are known. But most of the terms in the equation of light are unknown during reconstruction. The only known term is the viewing direction, **V**. Though it is possible to provide the lighting information, it is difficult to provide surface properties in the scene by only using camera images of the scene. Hence, to make the voxel-based IBR techniques more robust, there is a need for a mechanism that can predict variation in surface color due to different viewing orientations without any knowledge of surface properties.

3.2 The Color Cube Analysis

The Voxel Coloring [6] and the Generalized Voxel Coloring [1] algorithms are very much dependent on the color of the pixels in the images. So a specular highlight in any of the images can mislead the consistency evaluation test and hence cause the algorithm to fail. To understand how specular objects behave under varying illumination, we performed a *Color Cube Analysis* on synthetic and real datasets.

3.2.1 Synthetic Datasets

We used the OpenGL API to render a sphere with four different surface properties. These surface properties were defined in such a manner that it covered a range of real life scenarios:

- Low Diffuse (e.g., dull cloth)
- High Diffuse (e.g., bright painted walls)
- Low Specular (e.g., a white board or polished wood)
- High Specular (e.g., polished metallic surfaces)

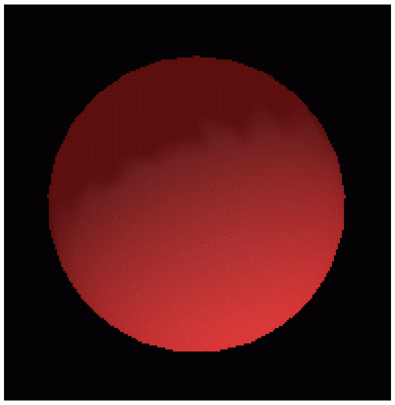
Figure 3.5: Sphere with low diffuse surface properties

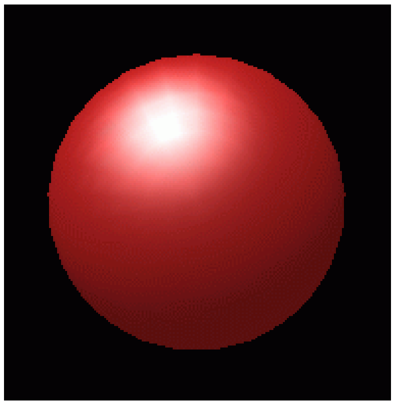
Figure 3.6: Sphere with bright specular highlight

Figure 3.5 shows a low diffuse sphere and Figure 3.6 shows a high specular sphere. Each surface property was rendered three times with red, green and blue color. Hence, we had twelve different spheres in all. Now for each one of these spheres, white light was flashed on the surface from 100 different uniformly distributed lighting positions. The color and intensity of light was kept constant throughout the experiment. For each lighting position, the color of each pixel of the surface was recorded. Finally, these observed colors were rendered onto the color cube.

Figure 3.7 shows low diffuse objects having a linear change in color from dark to bright, which is expected from the diffuse term in the equation of light (see Equation 3.6). As the angle between the source of light and the surface normal is decreased, the cosine term of the diffuse component adds more to the light reflection from the

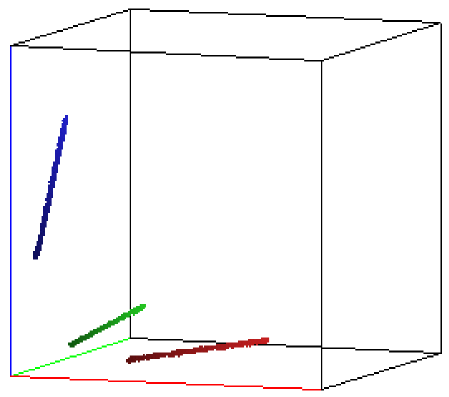

Figure 3.7: Pattern for low diffuse surface

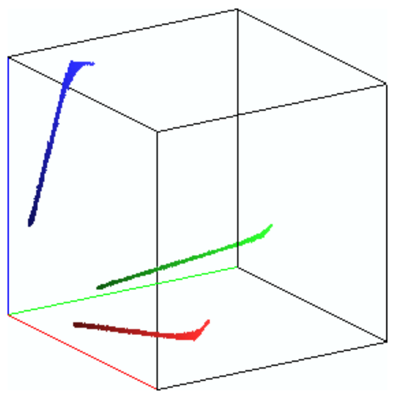

Figure 3.9: Pattern for low specular surface

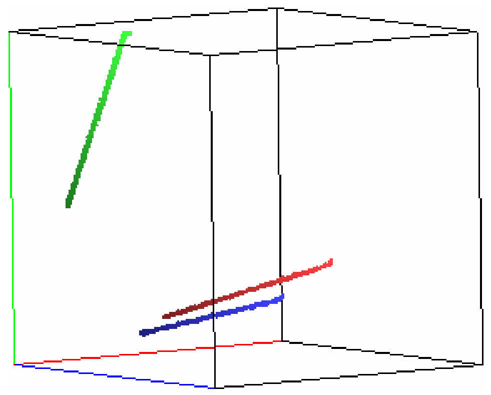

Figure 3.8: Pattern for high diffuse surface

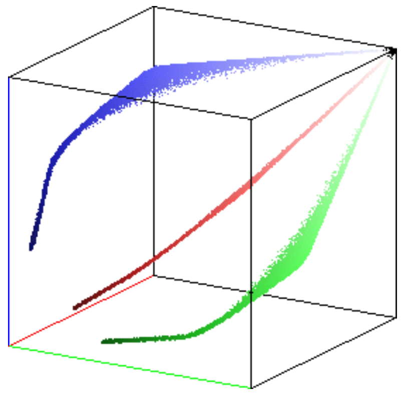

Figure 3.10: Pattern for high specular surface

surface and hence the linear straight pattern. For the second case of high diffuse objects, see Figure 3.8, we still observe a straight linear pattern. This is similar to what is obtained by the low diffuse, but is spread over a wider range of colors, simply because of a larger value of K_d. The pattern takes an interesting curve when we move to the third case of low specularity, Figure 3.9. Now, the specular component starts adding to the reflected color of light along with the diffuse component. The specular component is dependent on the viewing direction and the mirror direction. Hence, the light source adds color to the reflected light for certain viewing angles. This curvature, which is due to the specular component, is not as evident in the low specular case as it is in the final case of high specularity. For high specular surfaces, a complete banana shaped pattern is observed, see Figure 3.10, with the color ranging from the ambient color for the object to the color of light.

One of the inferences that we can draw from the patterns obtained is that the color of an object varies significantly depending on the orientation of the light source. Note that this explains why the conventional consistency evaluation algorithms fail. It can also be inferred that there exists a defined pattern for each of the material properties.

Also observe that in all color cubes, the reflected light is bounded by the ambient color of the surface and the color of light. For diffuse objects, the variation in reflected color is small and linear, but for specular objects a large variation in color is observed and these curves are not just straight lines, but they bend towards the color of light.

3.2.2 Real Datasets

Besides synthetic data, we ran the experiment on real scene data. Real datasets give a better picture of how the color varies under different lighting conditions.

To perform the experiment for real datasets, clips of images from different view-points were taken which had specular surfaces. Then the color values form these clips were rendered onto the color cube. Figure 3.13 shows the pattern obtained for polished wood edge shown in Figure 3.11 and Figure 3.14 shows the pattern for silver painted panel of a boom-box shown in Figure 3.12. The wooden material shows the typical banana curve for specular objects. On the other hand, the silver panel shows a very wide range of colors, but we do not see a bend in the curve because the color has equal components of red, green and blue. Hence, both cases show a substantial variation in color, which was expected from our observations about the synthetic data.

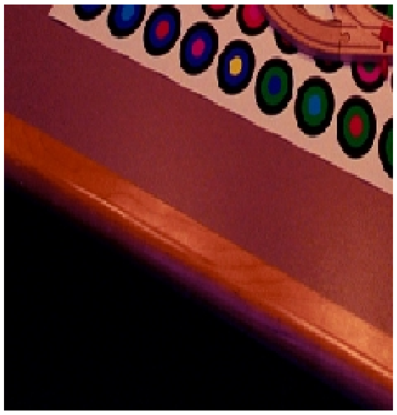
Figure 3.11: Polished wooden edge with specular highlights

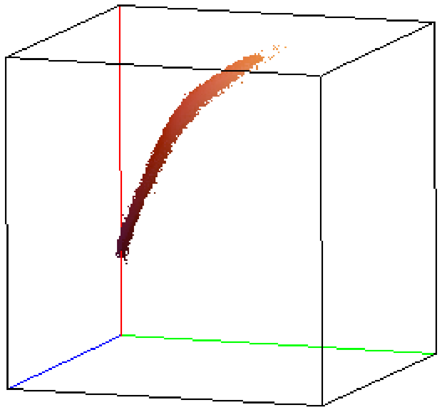
Figure 3.13: Pattern for polished wooden surface

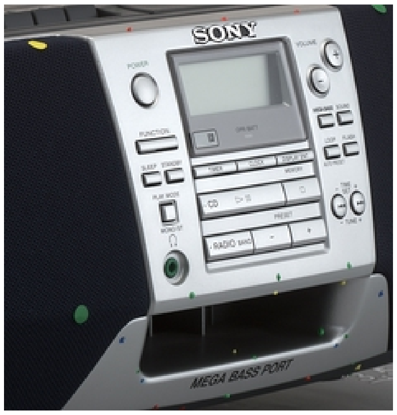

Figure 3.12: Silver painted panel of a boom-box shows large variation in color

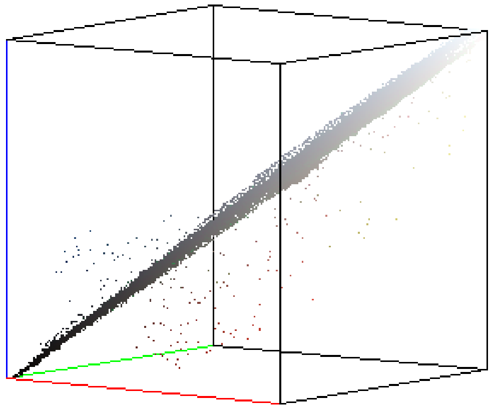
Figure 3.14: Pattern for silver painted surface

3.3 Problem Definition

Conventional consistency techniques rely on the variation in pixel colors of the images taken from different camera positions. Voxel based methods [6, 3, 1] determine consistency by putting an upperbound on the standard deviation in the pixel color among different images. But from our color cube analysis, we have shown that the same surfaces can show a wide color variation under different light orientations, especially for specular objects. If we can determine these curves (such as Figure 3.13) then our problem can be defined as: Given *a priori* curve, what is the probability that the two colors belong to the same surface:

$$\mathcal{P}\ (C_1, C_2 \in \text{same surface} \mid \textit{a priori}\ \text{curve})$$

Hence, to reconstruct objects that show a wide range of colors under different light orientations, we have to design a consistency check that is capable of predicting the possible range of colors the surface can reflect. At the same time, it should be robust enough not to pick up noise, which is possible since two different surfaces might be of different shades of the same color. In the next chapter, we present our consistency check that can predict this variation in color and reconstruct specular objects.

Chapter 4

Methodology

In chapter two, we discussed the Generalized Voxel Coloring (GVC) [1] which is an extension of the Voxel Coloring [6] but without any constraints on the camera positions. In our approach, we extended the GVC to reconstruct specular objects. This involved designing a new consistency technique which could predict the change in color of specular surfaces due to variations in light orientations. Apart from the consistency check mechanism, the core GVC algorithm was kept the same.

GVC begins with a volume, which bounds the scene. This volume is divided into voxels. Each voxel is projected onto all images (the voxel may not be visible in all the images) and the pixel colors are recorded. If the variation in the color of this set of pixels is too high then it is discarded or carved from the volume. This changes the visibility of voxels under the carved voxel. The decision whether to carve or not is made by the consistency test of the algorithm. The process is repeated until no more voxels are carved. When the algorithm is finished, all remaining surface voxels are consistent with the set of input images, and a 3D model of the scene is obtained. An essential role is played by the consistency test involved in this algorithm. A poor test can cause:

- Over-carving: unnecessary carving of voxels, which leads to a hollow or incomplete structure.
- Under-carving: not carving all the inconsistent voxels, hence generating a crude approximate volume.

In chapter three, we saw that specular surfaces show a wide range of colors. Also, we discussed why this variation causes the conventional techniques to fail. Now we present a new consistency check that deals better with specular objects.

4.1 Design

We designed a two-stage consistency check. The first stage reconstructs the Lambertian surfaces in the scene but fails to reconstruct surfaces with large variation in color. The second stage predicts the possible variation in the color of the surface due to change in viewing orientation and hence prevents the voxel from being carved.

We call our technique *Color Caching*. Like the GVC, we take a volume large enough to bound the scene. This volume is divided into voxels, whose size can be varied. We keep this voxel size large enough so that it projects to enough pixels to make a confident decision. All surface voxels are projected onto the images and the pixel colors are recorded but in a different manner. Conventional techniques usually maintain the sum for average or sum of squares for variance or a bin count in a discrete color space for each voxel, but we record the actual color. For each voxel, we maintain a cache for each view which is large enough to store all color values which each view can see. Hence unlike other techniques, we have the actual color values rather than some approximated representation. The cache size can be varied according to the needs of the dataset. To avoid wasting memory, each color entry is checked for repetitions before it is added to the cache. The coloring algorithm is shown in Figure 4.1.

```
for each voxel {
   define 'n' caches, where 'n' is the number of images
   for each view {
      C = color projected on this image
      if (colorExistsInCache(C) == false)
         add color to its cache
      }
}
```

Figure 4.1: Pseudo code for coloring the voxels

Once the coloring of the voxels is complete, each surface voxel is tested for consistency. The first pass divides the color space into small spheres and tries to find just one match in all possible combinations of views in which the voxel is visible. So for each voxel, all pairs of caches are exhaustively and if there exists a combination with at least one match then the voxel is consistent (as shown in Figure 4.2). If there is any pair of views for which there is no match then the voxel declared as inconsistent. A match for two colors is defined in terms of the thresholded Euclidian distance between the two colors:

$$\delta_{r_{i,j}} = red_{cache_i} - red_{cache_j} \tag{4.1}$$

$$\delta_{g_{i,j}} = green_{cache_i} - green_{cache_j} \tag{4.2}$$

$$\delta_{b_{i,j}} = blue_{cache_i} - blue_{cache_j} \tag{4.3}$$

$$\Delta_{i,j} = \sqrt{\delta^2_{r_{i,j}} + \delta^2_{g_{i,j}} + \delta^2_{b_{i,j}}} \tag{4.4}$$

$$\forall_i \, \forall_j \; if \; \exists \; \Delta_{i,j} \leq threshold, \; where \; i \neq j, \;\; \Rightarrow \;\; voxel \; is \; consistent \tag{4.5}$$

This stage is sensitive to color variation, since the threshold is not large enough

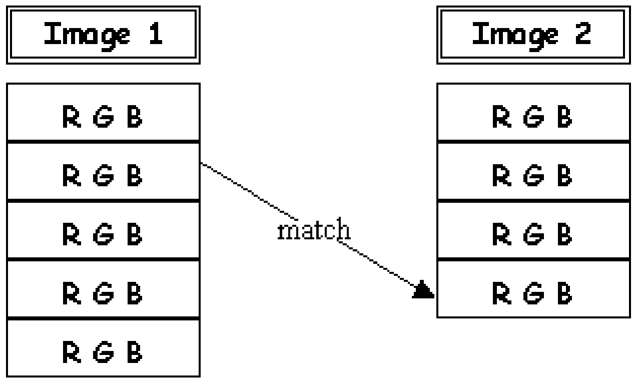

Figure 4.2: Cache comparison: just one match between each cache is required to declare a voxel as consistent

```
for each voxel {
   for each cache {
      if (there exist no match with any other cache) {
          voxel is inconsistent
          return
      }
   }
   voxel is consistent
}
```

Figure 4.3: Pseudo code for stage one consistency check

to cover a wide range of colors the surface could reflect. If the threshold is made large, then most of the voxels will be declared as consistent and we will obtain a very noisy reconstruction with lots of unwanted voxels. Hence, stage one can handle only Lambertian surfaces. The stage one consistency pseudo code is shown in Figure 4.3.

If stage one declares any pair of cache as inconsistent, the pair is tested by stage two before the voxel carved. This pair of cache has been declared inconsistent because variation in color among different views is beyond the defined threshold. This could either be due to the voxel really being inconsistent or there was a specular highlight in some of its views. We have seen that the change in color due to different lighting conditions shows a stable pattern. If we could predict this curve for each voxel, then we could prevent voxels from being carved due to specular highlight. But there are a number of unknowns in the equation of light: surface material parameters, surface normal and lighting information. This prevents us from determining the exact curve. However, for all practical purposes, this curve can be approximated by a straight line and still accurately predict specular highlights. If the two colors being compared lie on the same line within a permissible threshold, then we can conclude that the voxel is consistent.

From the equation of light (Equation 4.6), we know that the color reflected from the surface of an object depends on the surface properties and lighting parameters.

$$I = I_a K_a + I_i(K_d(\mathbf{L} \cdot \mathbf{N}) + K_s(\mathbf{R} \cdot \mathbf{V})^n) \tag{4.6}$$

$$\mathbf{R} = 2(\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L} \tag{4.7}$$

where

$\mathbf{L}$: Vector pointing to the direction of light

$\mathbf{N}$: Surface normal at the point of consideration

$\mathbf{R}$: Mirror direction of the surface

$\mathbf{V}$: The viewing direction

I_a, I_i : Light properties

K_a, K_d, K_s : Surface properties

n : Shininess factor for the surface

The images used as input for the carving algorithms differ only in the viewing orientation. Hence, the only term that varies in the equation of light for any surface in the scene is the viewing vector $\mathbf{V}$. This suggests that only the specular component varies in different views for the surface region under consideration. We know that the dot product in the specular term varies from 1 to 0 as the angle between **R** and **V** varies from 0 to 90 degrees and bounded between the same range of 0 and 1 for angles outside 0 and 90 degrees. Hence, the red, green and blue specular component for any surface increases monotonically as the angle between **R** and **V** decreases monotonically and vice-versa. Since the varying term, **V**, is same for all three color components, the red, green and blue component for the surface vary by the same proportion.

We approximate the color cube curve by a line whose coordinates vary by the same proportion. A line that passes through the origin has zero intercept and is of the form $y = mx$. So both x and y vary by the same proportion. For instance, when x becomes two fold, y too becomes two fold. Hence, in three dimension, for a line that passes through the origin, all the three color components vary by the same proportion. Thus, while comparing two color values one point can be used to find the equation of the line and the shortest distance from the other point to this line can decide if the variation in color is due different lighting or not.

Instead of finding the equation of line and then computing the shortest distance to that line, we can simplify the test by considering the fact that the ratio of each coordinates remain the same. Hence, the ratio of the two reds should be same as the ratio of the greens and the ratio of the blues:

$$\gamma_r = \frac{r_1}{r_2} \tag{4.8}$$

$$\gamma_g = \frac{g_1}{g_2} \tag{4.9}$$

$$\gamma_b = \frac{b_1}{b_2} \tag{4.10}$$

If the two colors belong to the same surface but are displaced on the color curve due to lighting variation then the second point should lie on the line and the three ratios $(\gamma_r, \gamma_g, \gamma_b)$ should be the same or the difference between them should be small. We define this difference as:

$$\delta_{rg} = |\gamma_r - \gamma_g| \tag{4.11}$$

$$\delta_{gb} = |\gamma_g - \gamma_b| \tag{4.12}$$

$$\delta_{br} = |\gamma_b - \gamma_r| \tag{4.13}$$

Since, we approximate the curve by a line, and there is always some noise in the scene, so we define another threshold, known as *tolerance* that defines how much can the point deviate from the line. The second stage check is defined as follows:

$$\Delta = \sqrt{\delta_{rg}^2 + \delta_{gb}^2 + \delta_{br}^2} \tag{4.14}$$

$$\Delta \leq tolerance \Rightarrow voxel\ is\ consistent \tag{4.15}$$

$$\Delta > tolerance \Rightarrow voxel\ is\ inconsistent \tag{4.16}$$

If the two colors are far apart on the color curve, they are considered to be inconsistent by the first stage or other conventional consistency checks. But the second stage prevents the voxel from being carved because of variation in color due to different lighting orientations. It predicts the possible range of colors a surface can have based on the fact that the only term that varies in the equation of light is the viewing direction, $\mathbf{V}$ and hence the change in the color of surface is monotonic and proportional.

However, to avoid any noise or unwanted voxels due to the leniency in our second stage test, another verification step is added. The caches which were being compared across each other for a match in stage one are now compared within themselves to find the intra-cache variance. This is an extension of the first stage as explained in Equation 4.5 but with $i = j$ for both caches such that each entry in a cache is compared to all other entries in the same cache. This verifies that the distribution is consistent within itself and adds to the confidence in the test.

Once the algorithm finishes and the volume is reconstructed, the volume is passed through a filter to remove all floating voxels (noise) in the reconstruction to generate an improved output.

Chapter 5

Evaluation

In order to evaluate the performance of our Color Caching algorithm, we tested it on both real and synthetic scenes. We developed an evaluation mechanism to determine the degree of accuracy of reconstruction, which we discuss in the following section.

5.1 Evaluation Mechanism

Comparing the output for real datasets can be hard because there is no ground truth model. One way to compare them is to render an image from a view that was not used to reconstruct the volume and then see how well it matches the original image [1]. But there are two disadvantages to this. First, it can provide evaluation information only for certain part of the volume and second, this technique cannot be used if there are a limited number of input views available.

We use both real and synthetic datasets to evaluate our algorithm. We compare the reconstructed image to the actual images used during reconstruction for the real datasets. We developed an evaluation framework that can generate synthetic datasets with both the images and the ground truth model. The ground truth model is the perfect 3D model that the carving algorithm should generate in an ideal case.

We used this ground truth model to compare the reconstruction generated by our algorithm.

5.1.1 Metrics

We defined three metrics to evaluate our algorithm. The first two metrics evaluate the images rendered using the model generated by our GVC algorithm. The third metric evaluates the 3D model itself but only for synthetic datasets, because we do not have the ground truth model for real datasets. These metrics are as follows:

- The first metric compares the change in re-projection error with time as the volume is carved. For each iteration of the carving algorithm, the images from the input camera viewpoints are rendered from the current volume. Then these reproduced images are compared to the original images and the average color difference in the pixel values is calculated. This gives a measure of how fast and how well each iteration carves the volume till the remaining volume approaches its best shape.

- A mask image is created for each of the input images. The mask image contains the information that identifies the foreground and the background pixels in the actual image. The second metric finds the amount of noise in the rendered image by comparing it to the scene's mask. *Noise* is defined as the number of pixels that fall outside the object boundaries. GVC carves the background and reconstructs only the objects in the foreground. Hence any background pixels seen in the rendered image are considered as unwanted noise. Once the final volume is obtained, images from the original camera viewpoints are rendered and are compared to the mask images. Then all the pixels that fall outside on the background region are counted and this gives us a measure of the amount of noise in the reconstruction.

- Finally, the third metric compares the volume itself. The 3D data structure that we use for defining the volume has four attributes, the three color components of the voxel and its visibility information. A voxel can be in any of three states: it can either be visible and on surface, it can be carved and hence not visible, or it can be hidden by other voxels. Using this visibility information, the reconstructed volume is compared to the ground truth model and the following numbers are recorded: the number of surface voxels matched, the number of voxels that should have been carved but were not (under-carved voxels), the number of voxels that should not have been carved but were carved (over-carved voxels) and finally, the number of voxels that were carved as expected (correctly carved). These four counts are then used to compute the percentage of surface match and the percentage of noise and evaluate the volume reconstruction.

5.2 Results

Our IBR pipeline has a component-based mechanism with which different consistency check methods can be plugged in and used to carve the volume. We compared the results of our Color Cache consistency check with three other methods: the Original GVC statistical based approach [1], the bin-count based Histogram approach [7] and the Silhouette approach [9] that uses mask images to reconstruct models. Only the consistency check component was changed for comparing these four methods and rest of the IBR pipeline was kept same. The Color Caching and the Original GVC requires tuning of their respective threshold parameters. These threshold values were tuned and set to obtain the best possible reconstruction for each approach.

5.2.1 Real Dataset

Our real dataset was a boom-box made up of shiny plastic material whose images were taken from nine different viewpoints, covering views from all sides and the top. Some of these views had broad specular highlights and some were extremely dull, hence covering a wide range of colors. In particular, the right side of the boom-box, Figure 5.1, shows a large variation in color when seen at an angle from the front side, Figure 5.2.

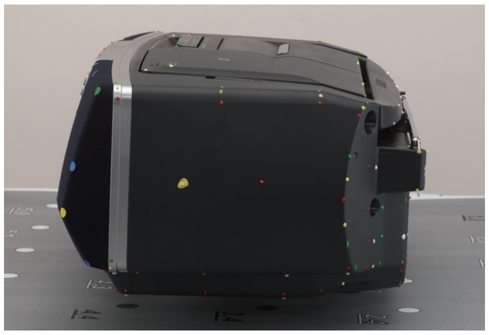
Figure 5.1: Boom-box: right view

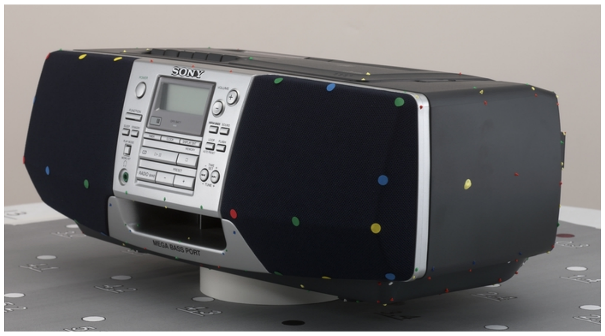
Figure 5.2: Boom-box: front-right view

We reconstructed the boom-box dataset using the four algorithms by plugging-in each one of the consistency checks at a time. The volumes obtained by these four approaches were filtered to remove unwanted floating voxels (noise). The filtered volumes for Color Cache, Histogram, Original GVC and Silhouette approach are shown in Figure 5.3, 5.4, 5.5 & 5.6 respectively.

Evidently, our Color Caching and the Silhouette are the only two approaches that are able to reconstruct the volume and do not fail for the specular highlighted region on the right side of the boom-box. The Silhouette reconstruction looks clear but it requires mask images as input, which identify the background pixels in the

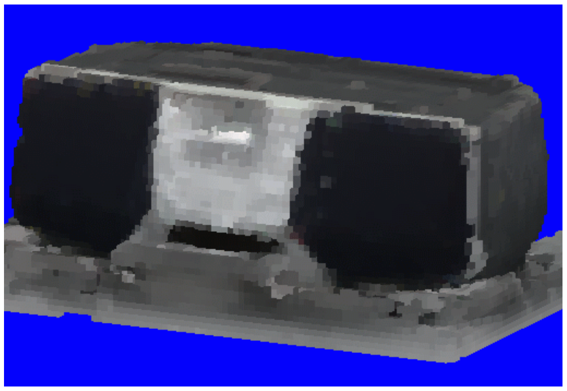
Figure 5.3: Color Cache Reconstruction

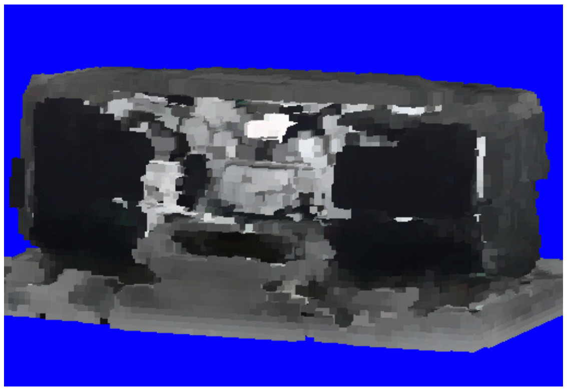
Figure 5.5: Original Reconstruction

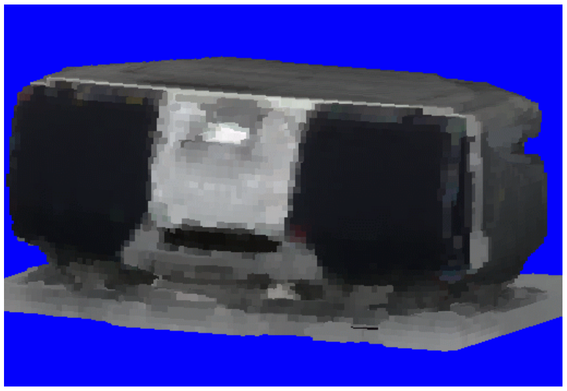
Figure 5.4: Histogram Reconstruction

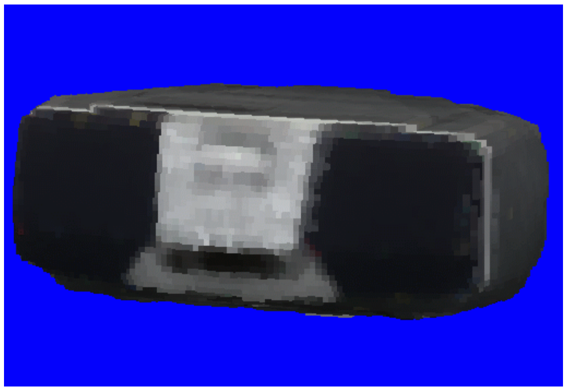
Figure 5.6: Silhouette Reconstruction

image. Creating mask images may not be easy for certain scenes, and moreover such techniques will fail for concave objects.

The Original GVC algorithm fails to reconstruct the volume due to high variation in color on the surface of the boom-box. The Histogram technique does a fairly good job, but fails to reconstruct the right side of the volume and over carves it as shown in Figure 5.7. Our Color Caching approach detects the variation in color due to different viewing orientation and reconstructs the right side of the boom-box as shown in Figure 5.8.

To show the working of our approach, we modified our algorithm to color all voxels that are carved by stage one as red, color all voxels that are carved by stage

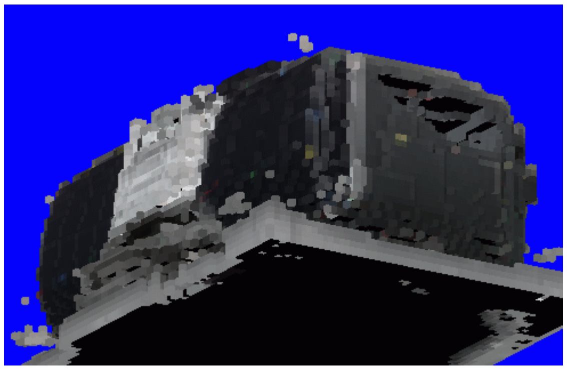

Figure 5.7: Histogram fails to reconstruct right side

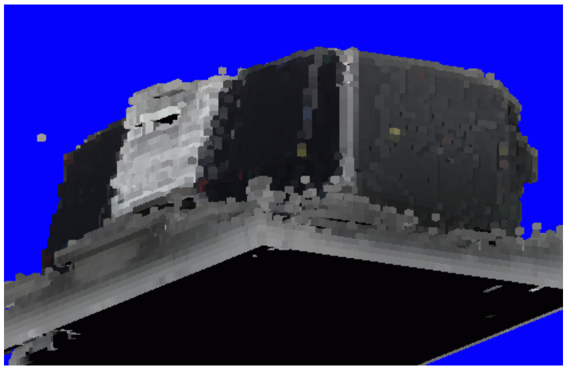

Figure 5.8: Color Caching detects color variation and reconstructs right side

two as green and color all voxels that were not carved due to insufficient input data as blue. As expected from our algorithm, the specular plastic body and the front panel of the boom-box are carved by stage two and the Lambertian black speakers are carved by stage one (as shown in Figure 5.9).

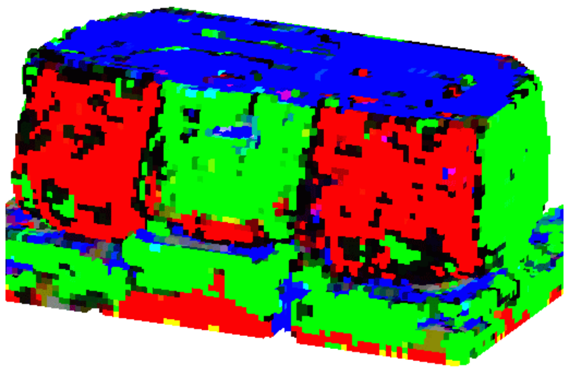

Figure 5.9: Color Caching reconstruction : red voxels are carved by stage one and green voxels are carved by stage two

To compare the performance of the four approaches for the boom-box dataset, we used our first metric: the re-projection error. The re-projection error for the first iteration is the same for Color Caching, Histogram and Original GVC, but is low for the Silhouette approach. This difference is due to the mask information available to the Silhouette approach with which it avoids the unwanted background voxels. The error decreases as the volume is carved. The error value at the final iteration eventually stabilizes with no significant difference between the four approaches. The Original GVC shows the least error, but it failed to reconstruct the boom-box scene. This suggests that this metric is not good enough to compare the different approaches. The metric works poorly because to compare the reconstructed image we used one of the images that was used to construct the volume. Ideally, an image that was not used in the input dataset should be used to compare the reconstruction quality.

However, this metric presents a good comparison of the the time taken by each approach to carve the volume. Figure 5.10 shows how the re-projection error drops with time for the front view of the boom-box. Our Color Caching technique takes the maximum amount of time, followed by the Original GVC, then the Histogram and finally the Silhouette. These results remain consistent for almost all views. Figure 5.11 shows the re-projection error for the front right side.

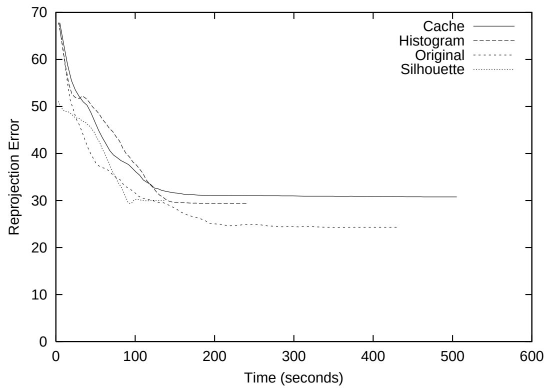

Figure 5.10: Convergence of the algorithms : front view of boom-box

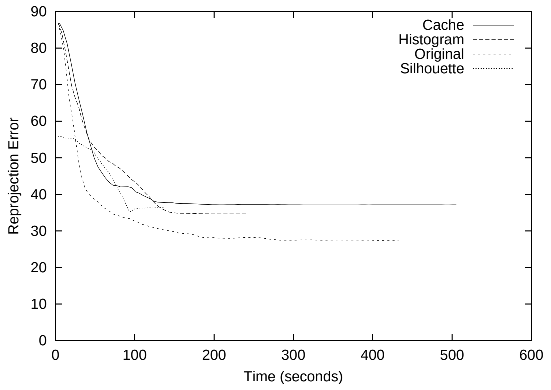

Figure 5.11: Convergence of the algorithms : front right view of boom-box

The amount of memory used by each algorithm also varies. Due to the large data structure maintained by the Color Caching technique, it occupies the largest amount

Algorithm	**Memory Usage**
Color Caching	104 MB
Histogram	62 MB
Original	48 MB
Silhouette	44 MB

Table 5.1: Memory usage comparison for different algorithms for boom-box dataset

of memory, followed by the Histogram, the Original GVC and the Silhouette. Table 5.1 shows the memory used for the boom-box dataset.

The noise in the rendered image also counts to the quality of reconstruction. A good reconstruction will have the least amount of noise. We used our second metric to compute the noise in the reconstructions generated by the four approaches. Figure 5.12 shows a comparison of noisy pixels count in the reconstruction for the front and the right view of the boom-box. Evidently, the Silhouette has the least amount of noise because it has the background information available from the mask, but interestingly, the other three algorithms have the same amount of noise, which shows that our Color Caching technique, despite of its leniency in color prediction, does not add any extra noise.

5.2.2 Synthetic Dataset

Shape is the most important factor in determining the quality of reconstruction. The re-projection error and the noise measure account only for the 2D perspective of the reconstruction and evaluate reconstruction quality only for the viewpoints that are visible in the set of input images. We used our third metric on a synthetic dataset to do a shape analysis, since we could generate the ground truth model for the synthetic dataset and compare it to the 3D data structure generated by the algorithm.

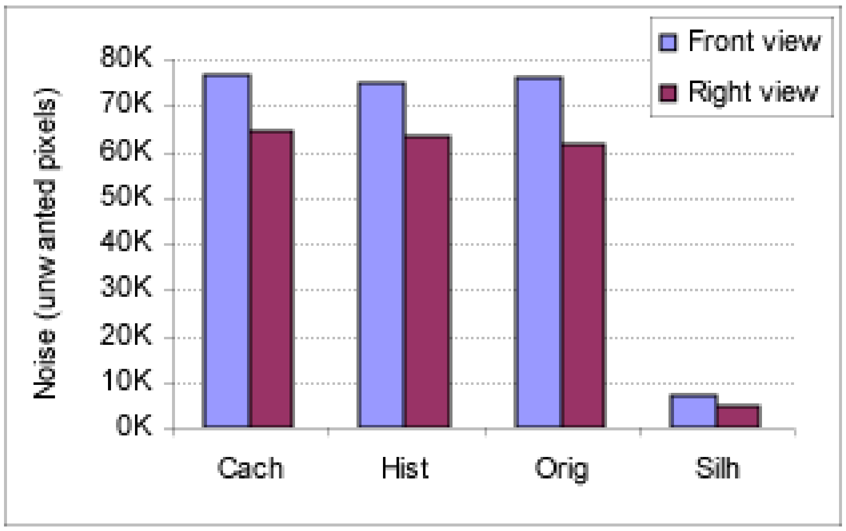

Figure 5.12: Noise in the rendered images for each algorithm for front and right view of boom-box

We developed an evaluation framework that can generate synthetic data sets and the equivalent ground truth model. The ground truth model is the perfect 3D shape of the object being constructed. This framework uses the OpenGL API to render geometrical figures, such as a sphere, from 12 different views, keeping lighting, surface properties and other parameters same. The viewpoints selected comprehensively cover the surface of the object such that while reconstruction the voxels are visible in more than one view. The ground truth model is generated using the equation of the object. The framework also outputs the camera parameters for each image, which is required as input along with the images to our carving algorithm.

Using our framework, we generated images of a sphere with bright specular highlight, Figure 5.13. We ran the carving algorithm with all four consistency techniques on this synthetic dataset. We measured the percentage surface match and the percentage noise for each reconstruction with respect to the ground truth model as:

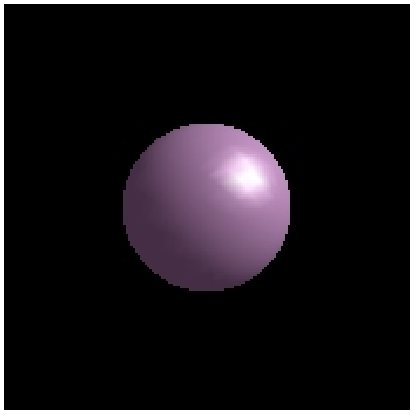

Figure 5.13: The synthetic dataset : a sphere with bright specular highlight

$$\%\ Surface\ Match = \frac{Number\ of\ surface\ voxels\ matched}{Total\ number\ of\ surface\ voxels} \times 100 \tag{5.1}$$

$$\%\ Noise = \frac{Number\ of\ undercarved\ voxels}{Number\ of\ voxels\ correctly\ carved} \times 100 \tag{5.2}$$

Each voxel can be in either of the three states: visible, hidden or carved. All the visible voxels in the ground truth model form the total number of surface voxels. Out of this all the voxels that are visible in the reconstructed model account to the surface match voxels and all the voxels that are carved in the reconstructed model are counted as over-carved voxels. The voxels that are visible in the reconstructed model but are carved in the ground truth model are considered as under-carved voxels.

Table 5.2 shows the results obtained for all the four approaches. Our Color Caching technique has the highest surface match with lowest amount of noise followed by Histogram, Figure 5.14 & 5.15 respectively. The Original GVC could not reconstruct the volume correctly, at high thresholds it left too much noise and at low

	Color Caching	**Histogram**	**Original GVC**	**Silhouette**
Good Match Voxels	4637	4604	4153	2479
Over Carved Voxels	339	372	823	2497
Total Surface Voxels	4976	4976	4976	4976
% Surface Match	**93.19 %**	**92.54 %**	**83.46 %**	**49.82 %**
Under Carved Voxels	1436	1478	2121	189
Correctly Carved Voxels	15924	15882	15239	17171
% Noise	**9.02 %**	**9.31 %**	**13.92 %**	**1.1 %**

Table 5.2: % Surface match and % noise for each reconstruction of sphere

threshold, it over-carved the volume near the specular highlighted region as shown in Figure 5.16. The Silhouette created a conservative model which though had very less noise but failed to meet the surface match criterion and carved a volume smaller than the actual size, Figure 5.17. This conservative model is generated because the Silhouette approach carves all the voxels with even a single pixel projecting onto the background.

5.3 Discussion

Our Color Caching approach successfully reconstructed scenes with specular surfaces. Our approach worked for both real and synthetic dataset. It is the only approach that completely reconstructed the boom-box scene without any knowledge of the surface properties or background pixels. We ran Color Caching algorithm a couple of times for tuning the *threshold* and the *tolerance* parameter to obtain the best possible reconstruction. The typical values for *threshold* vary between 25 to 35, and values for *tolerance* vary between 0.2 to 0.5. The Histogram performed as well as Color Caching for the synthetic sphere dataset. The Silhouette generated a conservative 3D model and the Original GVC failed to reconstruct specular objects.

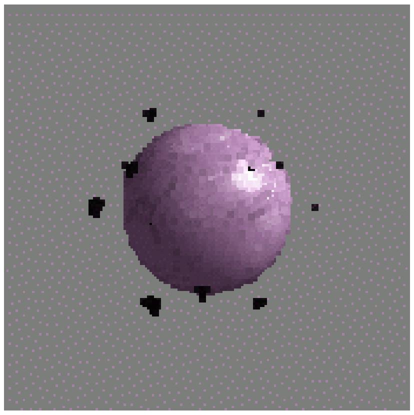

Figure 5.14: Sphere reconstructed by Color Caching

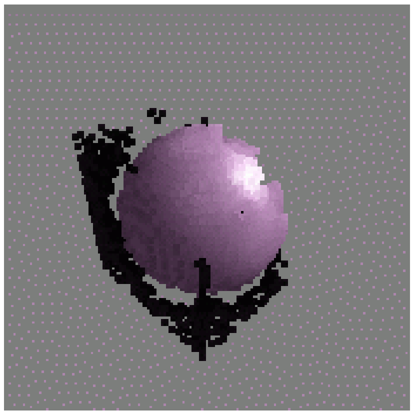

Figure 5.16: Sphere reconstructed by Original

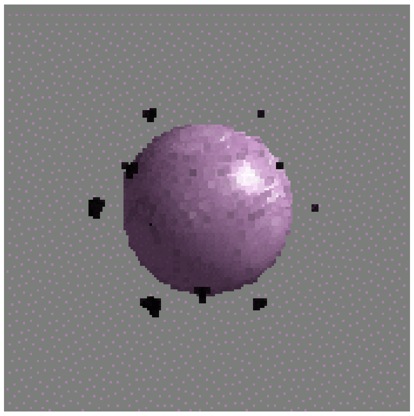

Figure 5.15: Sphere reconstructed by Histogram

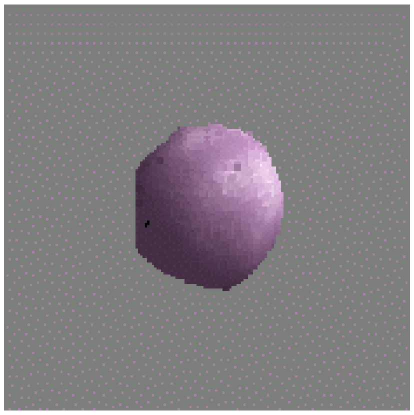

Figure 5.17: Sphere reconstructed by Silhouette

A comparison with other techniques shows that Color Caching takes the longest time to reconstruct the scenes and maximum amount of memory.

Chapter 6

Conclusions

Specular objects show a large variation in the color reflected from their surface under different viewing orientations. This variation in color exhibits a stable pattern both for real and synthetic datasets. The conventional consistency checks used in volumetric IBR approaches are sensitive to such high variation in color and hence fail to reconstruct specular scenes.

We have designed a new consistency evaluation mechanism that can predict the change in color of a surface due to changes in viewing orientation, irrespective of the surface material properties. We developed an evaluation mechanism and tested our technique on both real and synthetic datasets. Our Color Caching technique can reconstruct specular objects. It carves better volume structures, as compared to the Original GVC technique, for scenes with specular surfaces. The statistical deviation for surfaces with diffuse objects is lower than for surfaces with specularity. Since Original GVC places a common threshold on the standard deviation for the entire scene, Original GVC fails to reconstruct scenes with specular surfaces. The Histogram technique manages to reconstruct some specular scenes but fails when the variation in color is large. The Histogram fail for the extreme cases because if the bin size is small then the check fails to detect color variation and if the bin size is made too large then the reconstructions become noisy. A limitation of our Color Caching is that it requires two thresholds to be tuned for each scene, since it is a two-stage algorithm.

Due to the leniency in color prediction, our reconstruction leaves some unwanted voxels (noise) as compared to the Silhouette, which can be filtered after the volume is generated. But our approach does not place any constraint such as the requirement of mask images. The mask images required by the Silhouette approach can be difficult to create, and more over the Silhouette approach will work only for convex objects.

Color Caching uses the maximum amount of memory due to the large data structure maintained to store all colors for each view. The time taken by our technique is the highest. This suggests that there is scope for code optimization to make it run faster and use less memory.

Bibliography

[1] W. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *Proceedings of the ICCV Workshop, Vision Algorithms Theory and Practicee, Springer-Verlag Lecture Notes in Computer Science 1883*, pages 100–115, September 1999.

[2] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. *Computer Graphics*, 30(Annual Conference Series):303–312, 1996.

[3] K. N. Kutulakos and Steven Seitz. What do n photographs tell us about 3d shape? In *Technical Report TR680, Computer Science Dept., U. Rochester.* January 1998.

[4] A. Prock and C. Dyer. Towards real-time voxel coloring. In *In Proc. Image Understanding Workshop*, 1998.

[5] H. Saito and T. Kanade. Shape reconstruction in projective grid space from large number of images. In *In Proc. Computer Vision and Pattern Recognition Conference*, volume 2, pages 49–54, 1999.

[6] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. *Int. J. of Computer Vision*, 35(2):151–173, 1999.

[7] G. Slabaugh, W. Culbertson, T. Malzbender, M. Livingston, I. Sobel, M. Stevens, Lorentz, and R. Schafer. A collection of methods for volumetric reconstruction of visual scenes.

[8] Marc Soucy and Denis Laurendeau. A genral surface approach to the integration of a set of range views. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 17(4):344–358, 1995.

[9] Richard Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics, and Image Processing. Image Understanding*, 58(1):23–32, 1993.

[10] R. Tsai. A versatile camera calibration technique for high-accuracy 3-D machine vision metrology using off-the-shelf TV cameras and lenses. In L. Wolff,

S. Shafer, and G. Healey, editors, *Radiometry – (Physics-Based Vision)*. Jones and Bartlett, 1992.

[11] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. *Computer Graphics*, 28(Annual Conference Series):311–318, 1994.

[12] A. Watt and F. Policarpo. *The Computer Image*. ACM Press and SIGGRAPH Series, 1998.

[13] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. Technical Report 2927, Sophia-Antipolis Cedex, France, 1996.